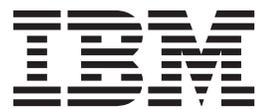


IBM DB2 10.1  
for Linux, UNIX, and Windows

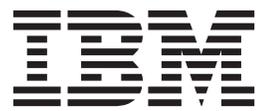
*Database Monitoring Guide and  
Reference*





IBM DB2 10.1  
for Linux, UNIX, and Windows

*Database Monitoring Guide and  
Reference*



**Note**

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 1557.

**Edition Notice**

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at <http://www.ibm.com/shop/publications/order>
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at <http://www.ibm.com/planetwide/>

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 2012.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

About this book . . . . . xxv

---

## Part 1. Interfaces for database monitoring . . . . . 1

Chapter 1. Database monitoring . . . . . 3

Chapter 2. Table functions for monitoring . . . . . 5

Monitoring system information using table functions . . . . . 5

Monitoring activities using table functions . . . . . 6

Monitoring data objects using table functions . . . . . 7

    Object usage . . . . . 8

Monitoring locking using table functions . . . . . 12

Monitoring system memory using table functions . . . . . 12

Other monitoring table functions . . . . . 13

Interfaces that return monitor data in XML documents. . . . . 13

    Interfaces for viewing XML monitor information as formatted text . . . . . 17

Chapter 3. Event monitors . . . . . 27

Types of events for which event monitors capture data . . . . . 27

Working with event monitors . . . . . 32

    Creating event monitors . . . . . 33

    Displaying a list of event monitors created in your database . . . . . 113

    Event monitors for partitioned databases and databases in a DB2 pureScale environment . . . . . 114

    Enabling event monitor data collection . . . . . 116

    Methods for accessing event monitor information . . . . . 118

    Altering an event monitor . . . . . 128

Monitoring different types of events. . . . . 129

    Lock and deadlock event monitoring . . . . . 129

    Unit of work event monitoring . . . . . 163

    Package cache statement eviction event monitoring . . . . . 216

    Activity event monitoring . . . . . 253

    Capturing system metrics using the statistics event monitor . . . . . 272

    Database event monitoring . . . . . 334

    Threshold violation event monitoring . . . . . 342

    Statement event monitoring . . . . . 343

    Table event monitoring . . . . . 348

    Buffer pool event monitoring . . . . . 350

    Table space event monitoring . . . . . 352

    Connection event monitoring . . . . . 354

    Transaction event monitoring . . . . . 360

    Deadlock event monitoring . . . . . 362

    Change history event monitoring . . . . . 366

Event monitor data retention from release to release. . . . . 403

Chapter 4. Other monitoring interfaces 405

Reports generated using the MONREPORT module 405

    Customizing the MONREPORT module reports 408

Snapshot monitor . . . . . 410

    Access to system monitor data: SYSMON authority . . . . . 411

    Capturing database system snapshots by using snapshot administrative views and table functions . . . . . 411

    Capturing database system snapshot information to a file using the

    SNAP\_WRITE\_FILE stored procedure . . . . . 414

    Accessing database system snapshots using snapshot table functions in SQL queries (with file access) . . . . . 416

    Snapshot monitor SQL Administrative Views 417

    SQL access to database system snapshots . . . . . 420

    Capturing a database snapshot from the CLP 421

    Snapshot monitor CLP commands . . . . . 422

    Capturing a database snapshot from a client application . . . . . 424

    Snapshot monitor API request types. . . . . 425

    Snapshot monitor sample output . . . . . 428

    Subsection snapshots . . . . . 429

    Global snapshots on partitioned database systems . . . . . 430

    Snapshot monitor self-describing data stream 431

    Monitoring with db2top in interactive mode commands . . . . . 433

Switch-based monitoring concepts . . . . . 438

    System monitor switches . . . . . 438

    Database system monitor data organization . . . . . 444

    Counter status and visibility . . . . . 445

    System monitor output: the self-describing data stream. . . . . 446

    Memory requirements for monitor data . . . . . 447

    Monitoring buffer pool activity . . . . . 449

    Database system monitor interfaces . . . . . 451

Determining the date a database object was last used . . . . . 453

Chapter 5. Deprecated monitoring tools . . . . . 455

Introduction to the health monitor . . . . . 455

    Health indicators . . . . . 455

    Enabling health alert notification . . . . . 488

    Health monitor . . . . . 490

Introduction to Windows Management

Instrumentation (WMI) . . . . . 510

    DB2 database system integration with Windows Management Instrumentation . . . . . 511

    Performance monitoring on Windows platforms 512

---

## Part 2. Monitor elements . . . . . 517

|  |            |
|--|------------|
| <b>Chapter 6. Request monitor elements</b>                                 | <b>519</b> |
| Activity monitor elements . . . . .  | 520        |
| <b>Chapter 7. Data object monitor elements</b>                             | <b>523</b> |
| <b>Chapter 8. Monitor element collection levels.</b>                       | <b>525</b> |
| <b>Chapter 9. Time-spent monitor elements</b>                              | <b>529</b> |
| Time-spent monitor element hierarchy . . . . .                             | 530        |
| Wait times for FCM communications . . . . .                                | 537        |
| Retrieving and working with time-spent monitor element data . . . . .      | 539        |
| Seeing where time is spent across the system                               | 539        |
| Determining where time is spent during SQL statement execution . . . . .   | 543        |
| <b>Chapter 10. Logical data groups overview</b>                            | <b>547</b> |
| Event monitor logical data groups and monitor elements . . . . .           | 547        |
| Event type mappings to logical data groups . . . . .                       | 596        |
| Logical data groups affected by COLLECT ACTIVITY DATA settings . . . . .   | 598        |
| Snapshot monitor interface mappings to logical data groups . . . . .       | 599        |
| Snapshot monitor logical data groups and monitor elements . . . . .        | 603        |
| <b>Chapter 11. Monitor element reference</b>                               | <b>635</b> |
| acc_curs_blk - Accepted Block Cursor Requests . . . . .                    | 636        |
| act_aborted_total - Total aborted activities monitor element . . . . .     | 636        |
| act_completed_total - Total completed activities monitor element . . . . . | 637        |
| act_cpu_time_top - Activity CPU time top monitor element . . . . .         | 638        |
| act_exec_time - Activity execution time monitor element . . . . .          | 639        |
| act_rejected_total - Total rejected activities monitor element . . . . .   | 640        |
| act_remapped_in - Activities remapped in monitor element . . . . .         | 641        |
| act_remapped_out - Activities remapped out monitor element . . . . .       | 641        |
| act_rows_read_top - Activity rows read top monitor element . . . . .       | 641        |
| act_rqsts_total - Total activity requests monitor elements . . . . .       | 642        |
| act_throughput - Activity throughput monitor element . . . . .             | 643        |
| act_total - Activities total monitor element. . . . .                      | 643        |
| activate_timestamp - Activate timestamp monitor element . . . . .          | 644        |
| active_hash_joins - Active hash joins . . . . .                            | 644        |
| active_olap_funcs - Active OLAP Functions monitor element . . . . .        | 644        |

|   |     |
|---|-----|
| active_sorts - Active Sorts . . . . .   | 644 |
| activity_collected - Activity collected monitor element . . . . .   | 645 |
| activity_id - Activity ID monitor element . . . . .   | 645 |
| activity_secondary_id - Activity secondary ID monitor element . . . . .                                   | 646 |
| activity_state - Activity state monitor element . . . . .   | 647 |
| activity_type - Activity type monitor element. . . . .  | 647 |
| activitytotaltime_threshold_id - Activity total time threshold ID monitor element . . . . .               | 648 |
| activitytotaltime_threshold_value - Activity total time threshold value monitor element . . . . .         | 648 |
| activitytotaltime_threshold_violated - Activity total time threshold violated monitor element . . . . .   | 649 |
| adapter_name - Adapter name monitor element   | 649 |
| address - IP address from which the connection was initiated . . . . .                                    | 649 |
| agent_id - Application handle (agent ID) monitor element . . . . .  | 650 |
| agent_id_holding_lock - Agent ID Holding Lock   | 651 |
| agent_pid - Engine dispatchable unit (EDU) identifier monitor element . . . . .                           | 652 |
| agent_status - DCS Application Agents. . . . .  | 652 |
| agent_sys_cpu_time - System CPU Time used by Agent . . . . .  | 653 |
| agent_tid - Agent thread ID monitor element. . . . .  | 653 |
| agent_usr_cpu_time - User CPU Time used by Agent . . . . .  | 653 |
| agent_wait_time - Agent wait time monitor element . . . . .   | 654 |
| agent_waits_total - Total agent waits monitor element . . . . .   | 655 |
| agents_created_empty_pool - Agents Created Due to Empty Agent Pool. . . . .                               | 656 |
| agents_from_pool - Agents Assigned From Pool  | 657 |
| agents_registered - Agents Registered . . . . .   | 657 |
| agents_registered_top - Maximum Number of Agents Registered . . . . .                                     | 658 |
| agents_stolen - Stolen Agents . . . . .   | 658 |
| agents_top - Number of Agents Created . . . . .   | 658 |
| agents_waiting_on_token - Agents Waiting for a Token . . . . .  | 659 |
| agents_waiting_top - Maximum Number of Agents Waiting monitor element . . . . .                           | 659 |
| agg_temp_tablespace_top - Aggregate temporary table space top monitor element . . . . .                   | 660 |
| aggsqltempespace_threshold_id - Aggregate SQL temporary space threshold ID monitor element . . . . .      | 660 |
| aggsqltempespace_threshold_value - AggSQL temporary space threshold value monitor element . . . . .       | 661 |
| aggsqltempespace_threshold_violated - AggSQL temporary space threshold violated monitor element . . . . . | 661 |
| app_act_aborted_total - Total failed external coordinator activities monitor element . . . . .            | 661 |
| app_act_completed_total - Total successful external coordinator activities monitor element . . . . .      | 662 |
| app_act_rejected_total - Total rejected external coordinator activities monitor element . . . . .         | 664 |
| appl_action - Application action monitor element  | 665 |

|  |     |  |     |
|--|-----|--|-----|
| app_rqsts_completed_total - Total application requests completed monitor element . . . . .     | 665 | bp_id - Buffer pool identifier monitor element . . . . .                         | 693 |
| appl_con_time - Connection Request Start Timestamp . . . . .                                   | 666 | bp_name - Buffer pool name monitor element . . . . .                             | 693 |
| appl_id - Application ID monitor element . . . . .   | 666 | bp_new_buffsz - New Buffer Pool Size . . . . .                                   | 694 |
| appl_id_holding_lk - Application ID Holding Lock   | 668 | bp_pages_left_to_remove - Number of Pages Left to Remove . . . . .               | 694 |
| appl_id_oldest_xact - Application with Oldest Transaction . . . . .                            | 669 | bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool . . . . .       | 694 |
| appl_idle_time - Application Idle Time . . . . .   | 670 | buff_auto_tuning - FCM buffer auto-tuning indicator monitor element . . . . .    | 695 |
| appl_name - Application name monitor element   | 670 | buff_free - FCM Buffers Currently Free . . . . .                                 | 695 |
| appl_priority - Application Agent Priority . . . . .   | 671 | buff_free_bottom - Minimum FCM Buffers Free                                      | 695 |
| appl_priority_type - Application Priority Type . . . . .                                       | 672 | buff_max - Maximum possible number of FCM buffers monitor element . . . . .      | 696 |
| appl_section_inserts - Section Inserts monitor element . . . . .                               | 672 | buff_total - Number of currently allocated FCM buffers monitor element . . . . . | 696 |
| appl_section_lookups - Section Lookups . . . . .   | 673 | byte_order - Byte Order of Event Data . . . . .                                  | 697 |
| appl_status - Application status monitor element   | 673 | cached_timestamp - Cached timestamp monitor element . . . . .                    | 697 |
| application_handle - Application handle monitor element . . . . .                              | 675 | cat_cache_inserts - Catalog cache inserts monitor element . . . . .              | 698 |
| apps_cur_cons - Applications Connected Currently   | 677 | cat_cache_lookups - Catalog cache lookups monitor element . . . . .              | 699 |
| apps_in_db2 - Applications Executing in the Database Currently . . . . .                       | 677 | cat_cache_overflows - Catalog Cache Overflows                                    | 700 |
| arm_correlator - Application response measurement correlator monitor element . . . . .         | 677 | cat_cache_size_top - Catalog cache high watermark monitor element . . . . .      | 701 |
| associated_agents_top - Maximum Number of Associated Agents . . . . .                          | 678 | catalog_node - Catalog Node Number . . . . .                                     | 702 |
| async_read_time - Asynchronous read time monitor element . . . . .                             | 678 | catalog_node_name - Catalog Node Network Name . . . . .                          | 702 |
| async_write_time - Asynchronous write time monitor element . . . . .                           | 678 | cf_waits - Number of cluster caching facility waits monitor element . . . . .    | 703 |
| async_runstats - Total number of asynchronous RUNSTATS requests monitor element . . . . .      | 678 | cf_wait_time - cluster caching facility wait time monitor element . . . . .      | 703 |
| audit_events_total - Total audit events monitor element . . . . .                              | 679 | cfg_collection_type - Configuration collection type                              | 704 |
| audit_file_write_wait_time - Audit file write wait time monitor element . . . . .              | 680 | cfg_name - Configuration name . . . . .  | 704 |
| audit_file_writes_total - Total audit files written monitor element . . . . .                  | 682 | cfg_old_value - Configuration old value . . . . .                                | 705 |
| audit_subsystem_wait_time - Audit subsystem wait time monitor element . . . . .                | 683 | cfg_old_value_flags - Configuration old value flags                              | 705 |
| audit_subsystem_waits_total - Total audit subsystem waits monitor element . . . . .            | 685 | cfg_value - Configuration value . . . . .  | 706 |
| auth_id - Authorization ID . . . . .   | 686 | cfg_value_flags - Configuration value flags . . . . .                            | 706 |
| authority_bitmap - User authorization level monitor element . . . . .                          | 687 | ch_auto_tuning - FCM channel auto-tuning indicator monitor element . . . . .     | 707 |
| authority_lvl - User authorization level monitor element . . . . .                             | 688 | ch_free - Channels Currently Free . . . . .                                      | 707 |
| auto_storage_hybrid - Hybrid automatic storage table space indicator monitor element . . . . . | 689 | ch_free_bottom - Minimum Channels Free . . . . .                                 | 707 |
| automatic - Buffer pool automatic monitor element  | 689 | ch_max - Maximum possible number of FCM channels monitor element . . . . .       | 708 |
| backup_timestamp - Backup timestamp . . . . .  | 689 | ch_total - Number of currently allocated FCM channels monitor element . . . . .  | 708 |
| bin_id - Histogram bin identifier monitor element  | 690 | client_acctng - Client accounting string monitor element . . . . .               | 709 |
| binds_precompiles - Binds/Precompiles Attempted  | 690 | client_applname - Client application name monitor element . . . . .              | 710 |
| block_ios - Number of block I/O requests monitor element . . . . .                             | 690 | client_db_alias - Database Alias Used by Application . . . . .                   | 711 |
| blocking_cursor - Blocking Cursor . . . . .  | 691 | client_hostname - Client hostname monitor element                                | 711 |
| blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element . . . . .           | 692 | client_idle_wait_time - Client idle wait time monitor element . . . . .          | 712 |
| bottom - Histogram bin bottom monitor element  | 692 | client_nname - Client name monitor element . . . . .                             | 713 |
| boundary_leaf_node_splits - Boundary leaf node splits monitor element . . . . .                | 693 | client_pid - Client process ID monitor element . . . . .                         | 713 |
| bp_cur_buffsz - Current Size of Buffer Pool . . . . .  | 693 | client_platform - Client operating platform monitor element . . . . .            | 714 |
|  |     | client_port_number - Client port number monitor element . . . . .                | 715 |

|  |     |  |     |
|--|-----|--|-----|
| client_prdid - Client product and version ID monitor element . . . . .   | 715 | concurrentdbcoordactivities_subclass_threshold_queued - Concurrent database coordinator activities service subclass threshold queued monitor element . . . . .           | 729 |
| client_protocol - Client communication protocol monitor element . . . . .  | 716 | concurrentdbcoordactivities_subclass_threshold_value - Concurrent database coordinator activities service subclass threshold value monitor element . . . . .             | 730 |
| client_userid - Client user ID monitor element . . . . .   | 717 | concurrentdbcoordactivities_subclass_threshold_violated - Concurrent database coordinator activities service subclass threshold violated monitor element . . . . .       | 730 |
| client_wrkstnname - Client workstation name monitor element . . . . .  | 717 | concurrentdbcoordactivities_superclass_threshold_id - Concurrent database coordinator activities service superclass threshold ID monitor element . . . . .               | 731 |
| codepage_id - ID of Code Page Used by Application . . . . .  | 718 | concurrentdbcoordactivities_superclass_threshold_queued - Concurrent database coordinator activities service superclass threshold queued monitor element . . . . .       | 731 |
| comm_exit_wait_time - Communication buffer exit wait time monitor element . . . . .  | 719 | concurrentdbcoordactivities_superclass_threshold_value - Concurrent database coordinator activities service superclass threshold value monitor element . . . . .         | 731 |
| comm_exit_waits - Communication buffer exit number of waits monitor element . . . . .  | 720 | concurrentdbcoordactivities_superclass_threshold_violated - Concurrent database coordinator activities service superclass threshold violated monitor element . . . . .   | 732 |
| comm_private_mem - Committed Private Memory  | 721 | concurrentdbcoordactivities_wl_was_threshold_id - Concurrent database coordinator activities workload work action set threshold ID monitor element . . . . .             | 732 |
| commit_sql_stmts - Commit Statements Attempted   | 721 | concurrentdbcoordactivities_wl_was_threshold_queued - Concurrent database coordinator activities workload work action set threshold queued monitor element . . . . .     | 733 |
| comp_env_desc - Compilation environment monitor element . . . . .  | 722 | concurrentdbcoordactivities_wl_was_threshold_value - Concurrent database coordinator activities workload work action set threshold value monitor element . . . . .       | 733 |
| completion_status - Completion status monitor element . . . . .  | 722 | concurrentdbcoordactivities_wl_was_threshold_violated - Concurrent database coordinator activities workload work action set threshold violated monitor element . . . . . | 734 |
| configured_cf_gbp_size - Configured cluster caching facility group buffer pool size monitor element . . . . .  | 723 | concurrentdbcoordactivities_work_action_set_threshold_id - Concurrent database coordinator activities work action set threshold ID monitor element . . . . .             | 734 |
| configured_cf_lock_size - Configured cluster caching facility lock size monitor element . . . . .  | 723 | concurrentdbcoordactivities_work_action_set_threshold_queued - Concurrent database coordinator activities work action set threshold queued monitor element . . . . .     | 734 |
| configured_cf_sca_size - Configured cluster caching facility shared communications area size monitor element . . . . .                                 | 723 | concurrentdbcoordactivities_work_action_set_threshold_value - Concurrent database coordinator activities work action set threshold value monitor element . . . . .       | 735 |
| configured_cf_mem_size - Configured cluster caching facility memory size monitor element . . . . .   | 724 | concurrentdbcoordactivities_work_action_set_threshold_violated - Concurrent database coordinator activities work action set threshold violated monitor element . . . . . | 735 |
| con_elapsed_time - Most Recent Connection Elapsed Time . . . . .   | 724 |  |     |
| con_local_dbases - Local Databases with Current Connects . . . . .   | 724 |  |     |
| con_response_time - Most Recent Response Time for Connect . . . . .  | 725 |  |     |
| concurrent_act_top - Concurrent activity top monitor element . . . . .   | 725 |  |     |
| concurrent_connection_top - Concurrent connection top monitor element . . . . .  | 726 |  |     |
| concurrent_wlo_act_top - Concurrent WLO activity top monitor element . . . . .   | 726 |  |     |
| concurrent_wlo_top - Concurrent workload occurrences top monitor element . . . . .   | 727 |  |     |
| concurrentdbcoordactivities_db_threshold_id - Concurrent database coordinator activities database threshold ID monitor element . . . . .               | 727 |  |     |
| concurrentdbcoordactivities_db_threshold_queued - Concurrent database coordinator activities database threshold queued monitor element . . . . .       | 728 |  |     |
| concurrentdbcoordactivities_db_threshold_value - Concurrent database coordinator activities database threshold value monitor element . . . . .         | 728 |  |     |
| concurrentdbcoordactivities_db_threshold_violated - Concurrent database coordinator activities database threshold violated monitor element . . . . .   | 729 |  |     |
| concurrentdbcoordactivities_subclass_threshold_id - Concurrent database coordinator activities service subclass threshold ID monitor element . . . . . | 729 |  |     |

|   |     |  |     |
|---|-----|--|-----|
| conn_complete_time - Connection Request Completion Timestamp . . . . .                                | 735 | cpu_cores_per_socket - Number of CPU cores per socket monitor element . . . . .                                  | 753 |
| conn_time - Time of database connection monitor element . . . . .                                     | 736 | cpu_hmt_degree - Number of logical CPUs monitor element . . . . .  | 753 |
| connection_start_time - Connection start time monitor element . . . . .                               | 736 | cpu_idle - Processor idle time monitor element . . . . .   | 753 |
| connection_status - Connection Status . . . . .   | 736 | cpu_iowait - IO Wait time monitor element . . . . .  | 754 |
| connections_top - Maximum Number of Concurrent Connections . . . . .                                  | 737 | cpu_limit - WLM dispatcher CPU limit monitor element . . . . .   | 755 |
| consistency_token - Package consistency token monitor element . . . . .                               | 738 | cpu_load_long - Processor load (long timeframe) monitor element . . . . .  | 755 |
| container_accessible - Accessibility of container monitor element . . . . .                           | 738 | cpu_load_medium - Processor load (medium timeframe) monitor element . . . . .                                    | 755 |
| container_id - Container identification monitor element . . . . .                                     | 739 | cpu_load_short - Processor load (short timeframe) monitor element . . . . .                                      | 756 |
| container_name - Container name monitor element . . . . .   | 739 | cpu_online - Number of CPUs online monitor element . . . . .   | 756 |
| container_stripe_set - Container stripe set monitor element . . . . .                                 | 739 | cpu_share_type - WLM dispatcher CPU share type monitor element . . . . .   | 756 |
| container_total_pages - Total pages in container monitor element . . . . .                            | 740 | cpu_shares - WLM dispatcher CPU shares monitor element . . . . .   | 756 |
| container_type - Container type monitor element . . . . .   | 740 | cpu_speed - CPU clock speed monitor element . . . . .  | 756 |
| container_usable_pages - Usable pages in container monitor element . . . . .                          | 741 | cpu_system - Kernel time monitor element . . . . .   | 757 |
| coord_act_aborted_total - Coordinator activities aborted total monitor element . . . . .              | 741 | cpu_timebase - Frequency of timebase register increment monitor element . . . . .                                | 757 |
| coord_act_completed_total - Coordinator activities completed total monitor element . . . . .          | 742 | cpu_total - Number of CPUs monitor element . . . . .   | 758 |
| coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element . . . . .        | 742 | cpu_usage_total - Processor usage monitor element . . . . .  | 758 |
| coord_act_exec_time_avg - Coordinator activities execution time average monitor element . . . . .     | 743 | cpu_user - Non-kernel processing time monitor element . . . . .  | 758 |
| coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element . . . . . | 744 | cpu_utilization - CPU utilization monitor element . . . . .  | 759 |
| coord_act_lifetime_avg - Coordinator activity lifetime average monitor element . . . . .              | 745 | cpu_velocity - CPU velocity monitor element . . . . .  | 760 |
| coord_act_lifetime_top - Coordinator activity lifetime top monitor element . . . . .                  | 746 | cputime_threshold_id - CPU time threshold ID monitor element . . . . .   | 760 |
| coord_agent_tid - Coordinator agent engine dispatchable unit ID monitor element . . . . .             | 747 | cputime_threshold_value - CPU time threshold value monitor element . . . . .                                     | 761 |
| coord_act_queue_time_avg - Coordinator activity queue time average monitor element . . . . .          | 747 | cputime_threshold_violated - CPU time threshold violated monitor element . . . . .                               | 761 |
| coord_act_rejected_total - Coordinator activities rejected total monitor element . . . . .            | 748 | cputimeinsc_threshold_id - CPU time in service class threshold ID monitor element . . . . .                      | 761 |
| coord_agent_pid - Coordinator agent identifier monitor element . . . . .                              | 748 | cputimeinsc_threshold_value - CPU time in service class threshold value monitor element . . . . .                | 762 |
| coord_agents_top - Maximum Number of Coordinating Agents . . . . .                                    | 749 | cputimeinsc_threshold_violated - CPU time in service class threshold violated monitor element . . . . .          | 762 |
| coord_member - Coordinator member monitor element . . . . .   | 749 | create_nickname - Create Nicknames . . . . .   | 762 |
| coord_node - Coordinating Node . . . . .  | 750 | create_nickname_time - Create Nickname Response Time . . . . .   | 763 |
| coord_partition_num - Coordinator partition number monitor element . . . . .                          | 750 | creator - Application Creator . . . . .  | 763 |
| coord_stmt_exec_time - Execution time for statement by coordinator agent monitor element . . . . .    | 751 | current_cf_gbp_size - Current cluster caching facility group buffer pool size monitor element . . . . .          | 764 |
| corr_token - DRDA Correlation Token . . . . .   | 751 | current_cf_lock_size - Current cluster caching facility lock size monitor element . . . . .                      | 764 |
| cost_estimate_top - Cost estimate top monitor element . . . . .                                       | 752 | current_cf_sca_size - Current cluster caching facility shared communications area size monitor element . . . . . | 764 |
| count - Number of Event Monitor Overflows . . . . .   | 752 | current_cf_mem_size - Current cluster caching facility memory size monitor element . . . . .                     | 764 |
| cpu_configured - Number of configured CPUs monitor element . . . . .                                  | 753 | current_active_log - Current Active Log File Number . . . . .  | 765 |
|   |     | current_archive_log - Current Archive Log File Number . . . . .  | 765 |
|   |     | current_extent - Extent currently being moved monitor element . . . . .  | 766 |

|  |     |   |     |
|--|-----|---|-----|
| current_request - Current operation request          |     | deferred - Deferred                                 | 785 |
| monitor element                                      | 766 | degree_parallelism - Degree of Parallelism          | 785 |
| cursor_name - Cursor Name                            | 766 | del_keys_cleaned - Pseudo deleted keys cleaned      |     |
| data_object_pages - Data Object Pages                | 766 | monitor element                                     | 786 |
| data_object_l_pages - Table data logical pages       |     | delete_sql_stmts - Deletes                          | 786 |
| monitor element                                      | 767 | delete_time - Delete Response Time                  | 786 |
| data_partition_id - Data partition identifier        |     | destination_service_class_id - Destination service  |     |
| monitor element                                      | 767 | class ID monitor element                            | 787 |
| datasource_name - Data Source Name                   | 768 | device_type - Device type                           | 787 |
| datataginsc_threshold_id - Data tag in service class |     | diaglog_write_wait_time - Diagnostic log file write |     |
| threshold (IN condition) ID                          | 769 | wait time monitor element                           | 788 |
| datataginsc_threshold_value - Data tag in service    |     | diaglog_writes_total - Total diagnostic log file    |     |
| class threshold (IN condition) value                 | 769 | writes monitor element                              | 789 |
| datataginsc_threshold_violated - Data tag in service |     | direct_read_reqs - Direct read requests monitor     |     |
| class threshold (IN condition) violated              | 769 | element   | 790 |
| datatagnotinsc_threshold_id - Data tag in service    |     | direct_read_time - Direct read time monitor         |     |
| class threshold (NOT IN condition) ID                | 770 | element   | 792 |
| datatagnotinsc_threshold_value - Data tag in         |     | direct_reads - Direct reads from database monitor   |     |
| service class threshold (NOT IN condition) value     | 770 | element   | 794 |
| datatagnotinsc_threshold_violated - Data tag in      |     | direct_write_reqs - Direct write requests monitor   |     |
| service class threshold (NOT IN condition) violated  | 770 | element   | 796 |
| db2_process_id - DB2 process ID monitor element      | 771 | direct_write_time - Direct write time monitor       |     |
| db2_process_name - DB2 process name monitor          |     | element   | 798 |
| element  | 771 | direct_writes - Direct writes to database monitor   |     |
| db2_status - Status of DB2 instance monitor          |     | element   | 800 |
| element  | 771 | disabled_peds - Disabled partial early distincts    |     |
| db2start_time - Start Database Manager Timestamp     | 772 | monitor element                                     | 802 |
| db_conn_time - Database activation timestamp         |     | disconn_time - Database Deactivation Timestamp      | 803 |
| monitor element                                      | 772 | disconnects - Disconnects                           | 803 |
| db_heap_top - Maximum Database Heap Allocated        | 773 | dl_conns - Connections involved in deadlock         |     |
| db_location - Database Location                      | 773 | monitor element                                     | 804 |
| db_name - Database name monitor element              | 773 | dynamic_sql_stmts - Dynamic SQL Statements          |     |
| db_path - Database Path                              | 774 | Attempted   | 804 |
| db_status - Status of database monitor element       | 775 | edu_id - Engine dispatchable unit ID monitor        |     |
| db_storage_path - Automatic storage path monitor     |     | element   | 805 |
| element  | 775 | eff_stmt_text - Effective statement text monitor    |     |
| db_storage_path_id - Storage path identifier         | 776 | element   | 805 |
| db_storage_path_state - Storage path state monitor   |     | effective_isolation - Effective isolation monitor   |     |
| element  | 776 | element   | 805 |
| db_storage_path_with_dpe - Storage path              |     | effective_lock_timeout - Effective lock timeout     |     |
| including database partition expression monitor      |     | monitor element                                     | 806 |
| element  | 777 | effective_query_degree - Effective query degree     |     |
| db_work_action_set_id - Database work action set     |     | monitor element                                     | 806 |
| ID monitor element                                   | 777 | elapsed_exec_time - Statement Execution Elapsed     |     |
| db_work_class_id - Database work class ID            |     | Time  | 806 |
| monitor element                                      | 778 | empty_pages_deleted - Empty pages deleted           |     |
| dbpartitionnum - Database partition number           |     | monitor element                                     | 807 |
| monitor element                                      | 778 | empty_pages_reused - Empty pages reused             |     |
| dcs_appl_status - DCS application status monitor     |     | monitor element                                     | 807 |
| element  | 780 | entry_time - Entry time monitor element             | 808 |
| dcs_db_name - DCS Database Name                      | 780 | estimated_cpu_entitlement - Estimated CPU           |     |
| ddl_classification - DDL classification              | 780 | entitlement monitor element                         | 808 |
| ddl_sql_stmts - Data Definition Language (DDL)       |     | estimatedsqlcost_threshold_id - Estimated SQL cost  |     |
| SQL Statements                                       | 781 | threshold ID monitor element                        | 808 |
| deadlock_id - Deadlock Event Identifier              | 782 | estimatedsqlcost_threshold_value - Estimated SQL    |     |
| deadlock_member - Deadlock member monitor            |     | cost threshold value monitor element                | 809 |
| element  | 782 | estimatedsqlcost_threshold_violated - Estimated     |     |
| deadlock_node - Partition Number Where               |     | SQL cost threshold violated monitor element         | 809 |
| Deadlock Occurred                                    | 783 | event_id - Event ID monitor element                 | 810 |
| deadlock_type - Deadlock type monitor element        | 783 | event_monitor_name - Event Monitor Name             | 811 |
| deadlocks - Deadlocks detected monitor element       | 783 | event_time - Event Time                             | 811 |

|   |     |  |     |
|---|-----|--|-----|
| event_timestamp - Event timestamp monitor element . . . . .                           | 811 | fcmtq_sends_total - FCM table queue send total monitor element . . . . .                     | 846 |
| event_type - Event Type monitor element . . . . .                                     | 812 | fetch_count - Number of Successful Fetches . . . . .   | 847 |
| evmon_activates - Number of event monitor activations . . . . .                       | 813 | files_closed - Database files closed monitor element . . . . .                               | 848 |
| evmon_wait_time - Event monitor wait time monitor element . . . . .                   | 814 | first_active_log - First Active Log File Number . . . . .                                    | 848 |
| evmon_waits_total - Event monitor total waits monitor element . . . . .               | 816 | first_overflow_time - Time of First Event Overflow . . . . .                                 | 849 |
| executable_id - Executable ID monitor element . . . . .                               | 818 | fs_caching - File system caching monitor element . . . . .                                   | 849 |
| executable_list_size - Size of executable list monitor element . . . . .              | 819 | fs_id - Unique file system identification number monitor element . . . . .                   | 850 |
| executable_list_truncated - Executable list truncated monitor element . . . . .       | 819 | fs_total_size - Total size of a file system monitor element . . . . .                        | 850 |
| evmon_flushes - Number of Event Monitor Flushes . . . . .                             | 819 | fs_used_size - Amount of space used on a file system monitor element . . . . .               | 851 |
| executable_id - Executable ID monitor element . . . . .                               | 820 | global_transaction_id - Global transaction identifier monitor element . . . . .              | 851 |
| execution_id - User Login ID . . . . .  | 820 | gw_comm_error_time - Communication Error Time . . . . .                                      | 852 |
| failed_sql_stmts - Failed Statement Operations . . . . .                              | 821 | gw_comm_errors - Communication Errors. . . . .   | 852 |
| fcmt_congested_sends - FCM congested sends monitor element . . . . .                  | 821 | gw_con_time - DB2 Connect Gateway First Connect Initiated . . . . .                          | 852 |
| fcmt_congestion_time - FCM congestion time monitor element . . . . .                  | 822 | gw_connections_top - Maximum Number of Concurrent Connections to Host Database. . . . .      | 852 |
| fcmt_num_congestion_timeouts - FCM congestion timeouts monitor element . . . . .      | 822 | gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request . . . . . | 853 |
| fcmt_num_conn_lost - FCM lost connections monitor element . . . . .                   | 822 | gw_cons_wait_host - Number of Connections Waiting for the Host to Reply . . . . .            | 853 |
| fcmt_num_conn_timeouts - FCM connection timeouts monitor element . . . . .            | 823 | gw_cur_cons - Current Number of Connections for DB2 Connect . . . . .                        | 854 |
| fcmt_message_rcv_volume - FCM message received volume monitor element . . . . .       | 823 | gw_db_alias - Database Alias at the Gateway. . . . .   | 854 |
| fcmt_message_rcv_wait_time - FCM message received wait time monitor element . . . . . | 824 | gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing . . . . .                | 854 |
| fcmt_message_rcvsv_total - Total FCM message receives monitor element . . . . .       | 826 | gw_total_cons - Total Number of Attempted Connections for DB2 Connect . . . . .              | 855 |
| fcmt_message_send_volume - FCM message send volume monitor element . . . . .          | 827 | hadr_connect_status - HADR Connection Status monitor element . . . . .                       | 855 |
| fcmt_message_send_wait_time - FCM message send wait time monitor element . . . . .    | 828 | hadr_connect_time - HADR Connection Time monitor element . . . . .                           | 856 |
| fcmt_message_sends_total - Total FCM message sends monitor element . . . . .          | 830 | hadr_heartbeat - HADR Heartbeat monitor element . . . . .                                    | 857 |
| fcmt_rcv_volume - FCM received volume monitor element . . . . .                       | 831 | hadr_local_host - HADR Local Host monitor element . . . . .                                  | 857 |
| fcmt_rcv_wait_time - FCM received wait time monitor element . . . . .                 | 832 | hadr_local_service - HADR Local Service monitor element . . . . .                            | 858 |
| fcmt_rcvsv_total - FCM receives total monitor element . . . . .                       | 834 | hadr_log_gap - HADR Log Gap . . . . .  | 858 |
| fcmt_send_volume - FCM send volume monitor element . . . . .                          | 835 | hadr_peer_window - HADR peer window monitor element . . . . .                                | 859 |
| fcmt_send_wait_time - FCM send wait time monitor element . . . . .                    | 836 | hadr_peer_window_end - HADR peer window end monitor element . . . . .                        | 859 |
| fcmt_sends_total - FCM sends total monitor element . . . . .                          | 838 | hadr_primary_log_file - HADR Primary Log File monitor element . . . . .                      | 860 |
| fcmt_tq_rcv_volume - FCM table queue received volume monitor element . . . . .        | 839 | hadr_primary_log_lsn - HADR Primary Log LSN monitor element . . . . .                        | 860 |
| fcmt_tq_rcv_wait_time - FCM table queue received wait time monitor element . . . . .  | 840 | hadr_primary_log_page - HADR Primary Log Page monitor element . . . . .                      | 861 |
| fcmt_tq_rcvsv_total - FCM table queue receives total monitor element . . . . .        | 842 | hadr_remote_host - HADR Remote Host monitor element . . . . .                                | 861 |
| fcmt_tq_send_volume - FCM table queue send volume monitor element . . . . .           | 843 | hadr_remote_instance - HADR Remote Instance monitor element . . . . .                        | 862 |
| fcmt_tq_send_wait_time - FCM table queue send wait time monitor element . . . . .     | 844 | hadr_remote_service - HADR Remote Service monitor element . . . . .                          | 862 |
|   |     | hadr_role - HADR Role . . . . .  | 863 |
|   |     | hadr_standby_log_file - HADR Standby Log File monitor element . . . . .                      | 863 |

|  |     |  |     |
|--|-----|--|-----|
| hadr_standby_log_lsn - HADR Standby Log LSN monitor element . . . . .                              | 864 | int_rows_inserted - Internal Rows Inserted . . . . .   | 884 |
| hadr_standby_log_page - HADR Standby Log Page monitor element . . . . .                            | 864 | int_rows_updated - Internal Rows Updated . . . . .   | 885 |
| hadr_state - HADR State monitor element . . . . .  | 864 | intra_parallel_state - Current state of intrapartition parallelism monitor element . . . . . | 886 |
| hadr_syncmode - HADR Synchronization Mode monitor element . . . . .                                | 865 | invocation_id - Invocation ID monitor element . . . . .                                      | 886 |
| hadr_timeout - HADR Timeout monitor element . . . . .  | 866 | ipc_rcv_volume - Interprocess communication received volume monitor element . . . . .        | 887 |
| hash_join_overflows - Hash Join Overflows . . . . .  | 867 | ipc_rcv_wait_time - Interprocess communication received wait time monitor element . . . . .  | 888 |
| hash_join_small_overflows - Hash Join Small Overflows . . . . .                                    | 867 | ipc_rcvs_total - Interprocess communication receives total monitor element . . . . .         | 889 |
| histogram_type - Histogram type monitor element . . . . .  | 868 | ipc_send_volume - Interprocess communication send volume monitor element . . . . .           | 890 |
| hld_application_handle - Identifier for the application holding the lock monitor element . . . . . | 869 | ipc_send_wait_time - Interprocess communication send wait time monitor element . . . . .     | 891 |
| hld_member - Database member for application holding lock . . . . .                                | 869 | ipc_sends_total - Interprocess communication send total monitor element . . . . .            | 892 |
| host_ccsid - Host Coded Character Set ID . . . . .   | 869 | is_system_appl - Is System Application monitor element . . . . .                             | 893 |
| host_db_name - Host Database Name . . . . .  | 870 | key_updates - Key updates monitor element . . . . .  | 893 |
| hostname - Host name monitor element . . . . .   | 870 | last_active_log - Last Active Log File Number . . . . .                                      | 893 |
| host_name - Host name monitor element . . . . .  | 871 | last_backup - Last Backup Timestamp . . . . .  | 894 |
| host_prdid - Host Product/Version ID . . . . .   | 871 | last_executable_id - Last executable identifier monitor element . . . . .                    | 894 |
| host_response_time - Host Response Time . . . . .  | 871 | last_extent - Last extent moved monitor element . . . . .                                    | 894 |
| id - cluster caching facility identification monitor element . . . . .                             | 872 | last_metrics_update - Metrics last update timestamp monitor element . . . . .                | 895 |
| idle_agents - Number of Idle Agents . . . . .  | 872 | last_overflow_time - Time of Last Event Overflow . . . . .                                   | 895 |
| iid - Index identifier monitor element . . . . .   | 873 | last_reference_time - Last reference time monitor element . . . . .                          | 895 |
| inbound_bytes_received - Inbound Number of Bytes Received . . . . .                                | 873 | last_request_type - Last request type monitor element . . . . .                              | 895 |
| inbound_bytes_sent - Inbound Number of Bytes Sent . . . . .  | 873 | last_reset - Last Reset Timestamp . . . . .  | 896 |
| inbound_comm_address - Inbound Communication Address . . . . .                                     | 874 | last_updated - Last update time stamp monitor element . . . . .                              | 897 |
| include_col_updates - Include column updates monitor element . . . . .                             | 874 | last_wlm_reset - Time of last reset monitor element . . . . .                                | 897 |
| incremental_bind - Incremental bind monitor element . . . . .                                      | 874 | lob_object_pages - LOB Object Pages . . . . .  | 898 |
| index_jump_scans - Index jump scans monitor element . . . . .                                      | 874 | lob_object_l_pages - LOB data logical pages monitor element . . . . .                        | 898 |
| index_name - Index name monitor element . . . . .  | 875 | local_cons - Local Connections . . . . .   | 899 |
| index_schema - Index schema monitor element . . . . .  | 875 | local_cons_in_exec - Local Connections Executing in the Database Manager . . . . .           | 899 |
| index_object_pages - Index Object Pages . . . . .  | 875 | local_start_time - Local start time monitor element . . . . .                                | 900 |
| index_object_l_pages - Index data logical pages monitor element . . . . .                          | 876 | local_transaction_id - Local transaction identifier monitor element . . . . .                | 900 |
| index_only_scans - Index-only scans monitor element . . . . .                                      | 876 | location - Location . . . . .  | 900 |
| index_scans - Index scans monitor element . . . . .  | 876 | location_type - Location type . . . . .  | 901 |
| index_tbsp_id - Index table space ID monitor element . . . . .                                     | 877 | lock_attributes - Lock attributes monitor element . . . . .                                  | 901 |
| input_db_alias - Input Database Alias . . . . .  | 877 | lock_count - Lock count monitor element . . . . .  | 902 |
| insert_sql_stmts - Inserts . . . . .   | 877 | lock_current_mode - Original lock mode before conversion monitor element . . . . .           | 903 |
| insert_time - Insert Response Time . . . . .   | 878 | lock_escalation - Lock escalation monitor element . . . . .                                  | 904 |
| insert_timestamp - Insert timestamp monitor element . . . . .                                      | 878 | lock_escals - Number of lock escalations monitor element . . . . .                           | 905 |
| int_auto_rebinds - Internal Automatic Rebinds . . . . .  | 879 | lock_escals_global - Number of global lock escalations monitor element . . . . .             | 907 |
| int_commits - Internal commits monitor element . . . . .   | 880 | lock_escals_locklist - Number of locklist lock escalations monitor element . . . . .         | 909 |
| int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock . . . . .                              | 881 | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element . . . . .         | 910 |
| int_node_splits - Intermediate node splits monitor element . . . . .                               | 882 |  |     |
| int_rollbacks - Internal rollbacks monitor element . . . . .                                       | 882 |  |     |
| int_rows_deleted - Internal Rows Deleted . . . . .   | 884 |  |     |

|  |     |   |     |
|--|-----|---|-----|
| lock_hold_count - Lock hold count monitor element . . . . .                          | 911 | long_tbsp_id - Long table space ID monitor element . . . . .  | 942 |
| lock_list_in_use - Total lock list memory in use monitor element . . . . .           | 912 | machine_identification - Host hardware identification monitor element. . . . .  | 942 |
| lock_mode - Lock mode monitor element . . . . .                                      | 912 | max_agent_overflows - Maximum Agent Overflows . . . . .   | 942 |
| lock_mode_requested - Lock mode requested monitor element . . . . .                  | 913 | max_coord_stmt_exec_time - Maximum coordinator statement execution time monitor element . . . . .                                   | 943 |
| lock_name - Lock name monitor element . . . . .                                      | 914 | max_coord_stmt_exec_time_args - Maximum coordinator statement execution time arguments monitor element . . . . .                    | 943 |
| lock_node - Lock Node . . . . .  | 915 | max_coord_stmt_exec_timestamp - Maximum coordinator statement execution timestamp monitor element . . . . .                         | 945 |
| lock_object_name - Lock Object Name . . . . .  | 915 | max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes . . . . .                     | 945 |
| lock_object_type - Lock object type waited on monitor element . . . . .              | 916 | max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes . . . . .                         | 946 |
| lock_release_flags - Lock release flags monitor element . . . . .                    | 918 | max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes . . . . .                  | 946 |
| lock_status - Lock status monitor element . . . . .                                  | 919 | max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes . . . . .                    | 947 |
| lock_timeout_val - Lock timeout value monitor element . . . . .                      | 920 | max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes . . . . .                       | 947 |
| lock_timeouts - Number of lock timeouts monitor element . . . . .                    | 920 | max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element . . . . . | 948 |
| lock_timeouts_global - Lock timeouts global monitor element . . . . .                | 922 | max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes . . . . .                    | 948 |
| lock_wait_end_time - Lock wait end timestamp monitor element . . . . .               | 923 | max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes . . . . .                       | 948 |
| lock_wait_start_time - Lock wait start timestamp monitor element . . . . .           | 923 | max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element . . . . . | 949 |
| lock_wait_time - Time waited on locks monitor element . . . . .                      | 924 | max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes . . . . .                    | 949 |
| lock_wait_time_global - Lock wait time global monitor element . . . . .              | 926 | max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes . . . . .                    | 950 |
| lock_wait_time_global_top - Top global lock wait time monitor element. . . . .       | 928 | max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes . . . . .                             | 950 |
| lock_wait_time_top - Lock wait time top monitor element . . . . .                    | 928 | max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes . . . . .                                 | 950 |
| lock_wait_val - Lock wait value monitor element . . . . .                            | 928 | max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes . . . . .                          | 951 |
| lock_waits - Lock waits monitor element . . . . .                                    | 928 | max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes . . . . .                            | 951 |
| lock_waits_global - Lock waits global monitor element . . . . .                      | 930 | max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes . . . . .                               | 952 |
| locks_held - Locks held monitor element . . . . .                                    | 932 |   |     |
| locks_held_top - Maximum number of locks held monitor element . . . . .              | 932 |   |     |
| locks_in_list - Number of Locks Reported . . . . .                                   | 933 |   |     |
| locks_waiting - Current agents waiting on locks monitor element . . . . .            | 933 |   |     |
| log_buffer_wait_time - Log buffer wait time monitor element . . . . .                | 933 |   |     |
| log_disk_wait_time - Log disk wait time monitor element . . . . .                    | 935 |   |     |
| log_disk_waits_total - Total log disk waits monitor element . . . . .                | 936 |   |     |
| log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages . . . . . | 937 |   |     |
| log_read_time - Log Read Time . . . . .  | 938 |   |     |
| log_reads - Number of Log Pages Read . . . . .                                       | 938 |   |     |
| log_to_redo_for_recovery - Amount of Log to be Redone for Recovery . . . . .         | 939 |   |     |
| log_write_time - Log Write Time . . . . .  | 940 |   |     |
| log_writes - Number of Log Pages Written . . . . .                                   | 940 |   |     |
| long_object_pages - Long Object Pages . . . . .                                      | 941 |   |     |
| long_object_l_pages - Long object data logical pages monitor element . . . . .       | 941 |   |     |

|   |     |  |     |
|---|-----|--|-----|
| max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes . . . . . | 952 | mon_interval_id - Monitor interval identifier monitor element . . . . .  | 967 |
| max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes . . . . .    | 953 | nesting_level - Nesting level monitor element . . . . .  | 967 |
| max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes . . . . .       | 953 | network_time_bottom - Minimum Network Time for Statement . . . . .   | 968 |
| max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes . . . . . | 954 | network_time_top - Maximum Network Time for Statement . . . . .  | 969 |
| max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes . . . . .    | 954 | nleaf - Number of leaf pages monitor element . . . . .   | 969 |
| max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes . . . . .    | 954 | nlevels - Number of index levels monitor element . . . . .   | 970 |
| max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms . . . . .            | 955 | no_change_updates - Number of no change row updates monitor element . . . . .                                  | 970 |
| max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms . . . . .               | 955 | node_number - Node Number . . . . .  | 970 |
| max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms . . . . .                      | 956 | nonboundary_leaf_node_splits - Non-boundary leaf node splits monitor element . . . . .                         | 971 |
| max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms . . . . .                 | 956 | num_agents - Number of Agents Working on a Statement . . . . .   | 971 |
| max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms . . . . .           | 957 | num_assoc_agents - Number of Associated Agents . . . . .   | 971 |
| max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms . . . . .            | 957 | num_compilations - Statement Compilations . . . . .  | 972 |
| member - Database member monitor element . . . . .  | 957 | num_coord_exec - Number of executions by coordinator agent monitor element . . . . .                           | 972 |
| memory_free - Amount of free physical memory monitor element . . . . .                                      | 961 | num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics monitor element . . . . . | 972 |
| memory_pool_used_hwm - Memory pool high water mark monitor element . . . . .                                | 961 | num_db_storage_paths - Number of automatic storage paths . . . . .   | 973 |
| memory_pool_id - Memory pool identifier monitor element . . . . .   | 961 | num_executions - Statement executions monitor element . . . . .  | 973 |
| memory_pool_type - Memory pool name monitor element . . . . .   | 962 | num_exec_with_metrics - Number of executions with metrics collected monitor element . . . . .                  | 974 |
| memory_pool_used - Amount of memory pool in use monitor element . . . . .                                   | 963 | num_extents_left - Number of extents left to process monitor element . . . . .                                 | 974 |
| memory_set_committed - Memory currently committed monitor element . . . . .                                 | 964 | num_extents_moved - Number of extents moved monitor element . . . . .  | 974 |
| memory_set_id - Memory set identifier monitor element . . . . .   | 964 | num_gw_conn_switches - Connection Switches . . . . .   | 974 |
| memory_set_size - Memory set size monitor element . . . . .   | 964 | num_indoubt_trans - Number of Indoubt Transactions . . . . .   | 975 |
| memory_set_type - Memory set type monitor element . . . . .   | 964 | num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element . . . . .          | 975 |
| memory_set_used - Memory in use by this set monitor element . . . . .                                       | 965 | num_log_data_found_in_buffer - Number of Log Data Found In Buffer. . . . .                                     | 977 |
| memory_set_used_hwm - Memory set high water mark monitor element . . . . .                                  | 965 | num_log_part_page_io - Number of Partial Log Page Writes . . . . .   | 977 |
| memory_swap_free - Total free swap space monitor element . . . . .  | 965 | num_log_read_io - Number of Log Reads . . . . .  | 978 |
| memory_swap_total - Total swap space monitor element . . . . .  | 966 | num_log_write_io - Number of Log Writes . . . . .  | 978 |
| memory_total - Total physical memory monitor element . . . . .  | 966 | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded monitor element . . . . .                     | 979 |
| message - Control Table Message. . . . .  | 966 | num_nodes_in_db2_instance - Number of Nodes in Partition . . . . .   | 980 |
| message_time - Timestamp Control Table Message . . . . .  | 967 | num_page_dict_built - Number of page-level compression dictionaries created or recreated. . . . .              | 980 |
|   |     | num_ref_with_metrics - Number of references with metrics monitor element. . . . .                              | 980 |
|   |     | num_references - Number of references monitor element . . . . .  | 981 |
|   |     | num_remaps - Number of remaps monitor element . . . . .  | 981 |
|   |     | num_tbsps - Number of table spaces monitor element . . . . .   | 981 |
|   |     | num_threshold_violations - Number of threshold violations monitor element . . . . .                            | 981 |
|   |     | num_transmissions - Number of Transmissions . . . . .  | 982 |

|   |     |  |      |
|---|-----|--|------|
| num_transmissions_group - Number of Transmissions Group . . . . .   | 983 | open_rem_curs_blk - Open Remote Cursors with Blocking . . . . .                      | 999  |
| number_in_bin - Number in bin monitor element   | 983 | os_level - Operating system level monitor element                                    | 1000 |
| object_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element. . . . .   | 983 | os_name - Operating system name monitor element . . . . .                            | 1000 |
| object_data_gbp_invalid_pages - GBP invalid data pages for a table monitor element . . . . .  | 984 | os_release - Operating system release monitor element . . . . .                      | 1000 |
| object_data_gbp_l_reads - GBP data logical reads for a table monitor element. . . . .   | 984 | os_version - Operating system version monitor element . . . . .                      | 1001 |
| object_data_gbp_p_reads - GBP data physical reads for a table monitor element. . . . .  | 985 | outbound_appl_id - Outbound Application ID   | 1001 |
| object_data_lbp_pages_found - LBP data pages found for a table monitor element . . . . .  | 986 | outbound_bytes_received - Outbound Number of Bytes Received . . . . .                | 1001 |
| object_data_l_reads - Buffer pool data logical reads for a table monitor element. . . . .   | 986 | outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received . . . . . | 1002 |
| object_data_p_reads - Buffer pool physical data reads for a table monitor element. . . . .  | 987 | outbound_bytes_received_top - Maximum Outbound Number of Bytes Received . . . . .    | 1002 |
| object_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element. . . . . | 987 | outbound_bytes_sent - Outbound Number of Bytes Sent . . . . .                        | 1002 |
| object_index_gbp_invalid_pages - GBP invalid index pages for an index monitor element. . . . .  | 988 | outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent . . . . .         | 1003 |
| object_index_gbp_l_reads - GBP index logical reads for an index monitor element . . . . .   | 988 | outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent . . . . .            | 1003 |
| object_index_gbp_p_reads - GBP index physical reads for an index monitor element . . . . .  | 989 | outbound_comm_address - Outbound Communication Address . . . . .                     | 1003 |
| object_index_lbp_pages_found - LBP index pages found for an index monitor element. . . . .  | 989 | outbound_comm_protocol - Outbound Communication Protocol . . . . .                   | 1004 |
| object_index_l_reads - Buffer pool index logical reads for an index monitor element . . . . .   | 990 | outbound_sequence_no - Outbound Sequence Number . . . . .                            | 1004 |
| object_index_p_reads - Buffer pool index physical reads for an index. . . . .   | 991 | overflow_accesses - Accesses to overflowed records monitor element . . . . .         | 1004 |
| object_name - Object name monitor element . . . . .   | 991 | overflow_creates - Overflow creates monitor element . . . . .                        | 1005 |
| object_requested - Requested object monitor element . . . . .   | 992 | package_id - Package identifier monitor element                                      | 1005 |
| object_schema - Object schema monitor element   | 992 | package_elapsed_time - Package elapsed time monitor element . . . . .                | 1005 |
| object_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element. . . . .     | 992 | package_list_count - Package list count monitor element . . . . .                    | 1006 |
| object_xda_gbp_invalid_pages - GBP invalid XDA data pages for a table monitor element . . . . .   | 993 | package_list_exceeded - Package list exceeded monitor element . . . . .              | 1006 |
| object_xda_gbp_l_reads - GBP XDA data logical read requests for a table monitor element . . . . .   | 993 | package_list_size - Size of package list monitor element . . . . .                   | 1006 |
| object_xda_gbp_p_reads - GBP XDA data physical read requests for a table monitor element . . . . .  | 994 | package_name - Package name monitor element  | 1006 |
| object_xda_lbp_pages_found - LBP XDA data pages found for a table monitor element . . . . .   | 995 | package_schema - Package schema monitor element . . . . .                            | 1007 |
| object_xda_l_reads - Buffer pool XDA data logical reads for a table monitor element. . . . .  | 995 | package_version_id - Package version monitor element . . . . .                       | 1007 |
| object_xda_p_reads - Buffer pool XDA data physical reads for a table monitor element. . . . .   | 996 | packet_receive_errors - Packet receive errors monitor element . . . . .              | 1008 |
| objtype - Object type monitor element . . . . .   | 996 | packets_received - Packets received monitor element . . . . .                        | 1008 |
| olap_func_overflows - OLAP Function Overflows monitor element . . . . .   | 997 | packet_send_errors - Packet send errors monitor element . . . . .                    | 1009 |
| open_cursors - Number of Open Cursors . . . . .   | 998 | packets_sent - Packets sent monitor element. . . . .                                 | 1009 |
| open_loc_curs - Open Local Cursors. . . . .   | 998 | page_allocations - Page allocations monitor element . . . . .                        | 1009 |
| open_loc_curs_blk - Open Local Cursors with Blocking . . . . .  | 998 | page_reorgs - Page reorganizations monitor element . . . . .                         | 1009 |
| open_rem_curs - Open Remote Cursors . . . . .   | 999 | page_reclaims_x - Page reclaims exclusive access monitor element . . . . .           | 1010 |
|   |     | page_reclaims_s - Page reclaims shared access monitor element . . . . .              | 1010 |

|   |      |   |      |
|---|------|---|------|
| page_reclaims_initiated_x - Page reclaims initiated exclusive access monitor element . . . . .  | 1011 | pool_async_data_reads - Buffer pool asynchronous data reads monitor element . . . . .   | 1027 |
| page_reclaims_initiated_s - Page reclaims initiated shared access monitor element . . . . .   | 1011 | pool_async_data_writes - Buffer pool asynchronous data writes monitor element . . . . .   | 1028 |
| pages_from_block_ios - Total number of pages read by block I/O monitor element. . . . .   | 1011 | pool_async_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element . . . . .                 | 1029 |
| pages_from_vectored_ios - Total number of pages read by vectored I/O monitor element . . . . .  | 1012 | pool_async_index_gbp_invalid_pages - Asynchronous group buffer pool invalid index pages monitor element . . . . .   | 1029 |
| pages_merged - Pages merged monitor element   | 1012 | pool_async_index_gbp_l_reads - Asynchronous group buffer pool index logical reads monitor element . . . . .   | 1030 |
| pages_read - Number of pages read monitor element . . . . .   | 1013 | pool_async_index_gbp_p_reads - Asynchronous group buffer pool index physical reads monitor element . . . . .  | 1030 |
| pages_written - Number of pages written monitor element . . . . .   | 1013 | pool_async_index_lbp_pages_found - Asynchronous local buffer pool index pages found monitor element. . . . .  | 1031 |
| parent_activity_id - Parent activity ID monitor element . . . . .   | 1013 | pool_async_index_read_reqs - Buffer pool asynchronous index read requests monitor element . . . . .   | 1031 |
| parent_uow_id - Parent unit of work ID monitor element . . . . .  | 1014 | pool_async_index_reads - Buffer pool asynchronous index reads monitor element . . . . .   | 1032 |
| partial_record - Partial Record monitor element   | 1014 | pool_async_index_writes - Buffer pool asynchronous index writes monitor element. . . . .  | 1033 |
| participant_no - Participant within Deadlock . . . . .  | 1015 | pool_async_read_time - Buffer Pool Asynchronous Read Time . . . . .   | 1033 |
| participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application. . . . .   | 1015 | pool_async_write_time - Buffer pool asynchronous write time monitor element . . . . .   | 1034 |
| participant_type - Participant type monitor element . . . . .   | 1016 | pool_async_xda_gbp_indep_pages_found_in_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element . . . . . | 1035 |
| partition_key - Partitioning key monitor element  | 1016 | pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages monitor element . . . . .  | 1036 |
| partition_number - Partition Number . . . . .   | 1017 | pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests monitor element . . . . .  | 1036 |
| passthru_time - Pass-Through Time . . . . .   | 1017 | pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests monitor element . . . . .   | 1037 |
| passthru - Pass-Through . . . . .   | 1017 | pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found monitor element . . . . .  | 1037 |
| past_activities_wrapped - Past activities list wrapped monitor element. . . . .   | 1018 | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests monitor element   | 1038 |
| phase_start_event_id - Phase start event ID . . . . .   | 1018 | pool_async_xda_reads - Buffer pool asynchronous XDA data reads monitor element . . . . .  | 1038 |
| phase_start_event_timestamp - Phase start event timestamp . . . . .   | 1018 | pool_async_xda_writes - Buffer pool asynchronous XDA data writes monitor element . . . . .  | 1039 |
| pipedsorts_accepted - Piped Sorts Accepted . . . . .  | 1019 | pool_config_size - Configured Size of Memory Pool . . . . .   | 1040 |
| pipedsorts_requested - Piped Sorts Requested  | 1019 | pool_cur_size - Current Size of Memory Pool   | 1041 |
| pkg_cache_inserts - Package cache inserts monitor element . . . . .   | 1020 | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element . . . . .  | 1041 |
| pkg_cache_lookups - Package cache lookups monitor element . . . . .   | 1021 |   |      |
| pkg_cache_num_overflows - Package Cache Overflows . . . . .   | 1023 |   |      |
| pkg_cache_size_top - Package cache high watermark . . . . .   | 1023 |   |      |
| pool_async_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element . . . . . | 1024 |   |      |
| pool_async_data_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element . . . . .   | 1024 |   |      |
| pool_async_data_gbp_l_reads - Asynchronous group buffer pool data logical reads monitor element . . . . .   | 1025 |   |      |
| pool_async_data_gbp_p_reads - Asynchronous group buffer pool data physical reads monitor element . . . . .  | 1025 |   |      |
| pool_async_data_lbp_pages_found - Asynchronous local buffer pool data pages found monitor element . . . . .   | 1026 |   |      |
| pool_async_data_read_reqs - Buffer pool asynchronous read requests monitor element . . . . .  | 1026 |   |      |

|  |      |  |      |
|--|------|--|------|
| pool_data_gbp_invalid_pages - Group buffer pool<br>invalid data pages monitor element . . . . .  | 1042 | pool_queued_async_data_reqs - Data prefetch<br>requests monitor element . . . . .  | 1089 |
| pool_data_gbp_l_reads - Group buffer pool data<br>logical reads monitor element . . . . .  | 1044 | pool_queued_async_index_pages - Index pages<br>prefetch requests monitor element . . . . .                                     | 1091 |
| pool_data_gbp_p_reads - Group buffer pool data<br>physical reads monitor element . . . . .   | 1045 | pool_queued_async_index_reqs - Index prefetch<br>requests monitor element . . . . .  | 1093 |
| pool_data_lbp_pages_found - Local buffer pool<br>found data pages monitor element . . . . .  | 1047 | pool_queued_async_other_reqs - Other requests<br>handled by prefetchers monitor element . . . . .                              | 1095 |
| pool_data_l_reads - Buffer pool data logical reads<br>monitor element . . . . .  | 1048 | pool_queued_async_temp_data_pages - Data<br>pages prefetch requests for temporary table spaces<br>monitor element . . . . .    | 1097 |
| pool_data_p_reads - Buffer pool data physical<br>reads monitor element . . . . .   | 1050 | pool_queued_async_temp_data_reqs - Data<br>prefetch requests for temporary table spaces<br>monitor element . . . . .           | 1099 |
| pool_data_writes - Buffer pool data writes<br>monitor element . . . . .  | 1052 | pool_queued_async_temp_index_pages - Index<br>pages prefetch requests for temporary table spaces<br>monitor element . . . . .  | 1101 |
| pool_drty_pg_steal_clns - Buffer pool victim page<br>cleaners triggered monitor element . . . . .  | 1055 | pool_queued_async_temp_index_reqs - Index<br>prefetch requests for temporary table spaces<br>monitor element . . . . .         | 1103 |
| pool_drty_pg_thrsh_clns - Buffer pool threshold<br>cleaners triggered monitor element . . . . .  | 1056 | pool_queued_async_temp_xda_pages - XDA data<br>pages prefetch requests for temporary table spaces<br>monitor element . . . . . | 1105 |
| pool_failed_async_data_reqs - Failed data prefetch<br>requests monitor element . . . . .   | 1057 | pool_queued_async_temp_xda_reqs - XDA data<br>prefetch requests for temporary table spaces<br>monitor element . . . . .        | 1107 |
| pool_failed_async_index_reqs - Failed index<br>prefetch requests monitor element . . . . .   | 1059 | pool_queued_async_xda_pages - XDA pages<br>prefetch requests monitor element . . . . .   | 1109 |
| pool_failed_async_other_reqs - Failed<br>non-prefetch requests monitor element . . . . .   | 1061 | pool_queued_async_xda_reqs - XDA prefetch<br>requests monitor element . . . . .  | 1111 |
| pool_failed_async_temp_data_reqs - Failed data<br>prefetch requests for temporary table spaces<br>monitor element . . . . .                        | 1063 | pool_read_time - Total buffer pool physical read<br>time monitor element . . . . .   | 1113 |
| pool_failed_async_temp_index_reqs - Failed index<br>prefetch requests for temporary table spaces<br>monitor element . . . . .                      | 1065 | pool_secondary_id - Memory Pool Secondary<br>Identifier . . . . .  | 1114 |
| pool_failed_async_temp_xda_reqs - Failed XDA<br>prefetch requests for temporary table spaces<br>monitor element . . . . .                          | 1067 | pool_sync_data_gbp_reads - Synchronous group<br>buffer pool data reads monitor element . . . . .                               | 1115 |
| pool_failed_async_xda_reqs - Failed XDA prefetch<br>requests monitor element . . . . .   | 1069 | pool_sync_data_reads - Synchronous buffer pool<br>data reads monitor element . . . . .   | 1115 |
| pool_id - Memory Pool Identifier . . . . .   | 1071 | pool_sync_index_gbp_reads - Synchronous group<br>buffer pool index reads monitor element . . . . .                             | 1115 |
| pool_index_gbp_indep_pages<br>_found_in_lbp - Group buffer pool independent<br>index pages found in local buffer pool monitor<br>element . . . . . | 1073 | pool_sync_index_reads - Synchronous buffer pool<br>index reads monitor element . . . . .                                       | 1116 |
| pool_index_gbp_invalid_pages - Group buffer<br>pool invalid index pages monitor element . . . . .  | 1074 | pool_sync_xda_gbp_reads - Synchronous group<br>buffer pool XDA data reads monitor element . . . . .                            | 1116 |
| pool_index_gbp_l_reads - Group buffer pool<br>index logical reads monitor element . . . . .  | 1075 | pool_sync_xda_reads - Synchronous buffer pool<br>XDA data reads monitor element . . . . .                                      | 1116 |
| pool_index_gbp_p_reads - Group buffer pool<br>index physical reads monitor elements . . . . .  | 1077 | pool_temp_data_l_reads - Buffer pool temporary<br>data logical reads monitor element . . . . .                                 | 1116 |
| pool_index_lbp_pages_found - Local buffer pool<br>index pages found monitor element . . . . .  | 1078 | pool_temp_data_p_reads - Buffer pool temporary<br>data physical reads monitor element . . . . .                                | 1118 |
| pool_index_l_reads - Buffer pool index logical<br>reads monitor element . . . . .  | 1080 | pool_temp_index_l_reads - Buffer pool temporary<br>index logical reads monitor element . . . . .                               | 1120 |
| pool_index_p_reads - Buffer pool index physical<br>reads monitor element . . . . .   | 1082 | pool_temp_index_p_reads - Buffer pool temporary<br>index physical reads monitor element . . . . .                              | 1121 |
| pool_index_writes - Buffer pool index writes<br>monitor element . . . . .  | 1084 | pool_temp_xda_l_reads - Buffer pool temporary<br>XDA data logical reads monitor element . . . . .                              | 1123 |
| pool_lsn_gap_clns - Buffer pool log space cleaners<br>triggered monitor element . . . . .  | 1086 | pool_temp_xda_p_reads - Buffer pool temporary<br>XDA data physical reads monitor element . . . . .                             | 1125 |
| pool_no_victim_buffer - Buffer pool no victim<br>buffers monitor element . . . . .   | 1087 | pool_watermark - Memory Pool Watermark . . . . .   | 1127 |
| pool_queued_async_data_pages - Data pages<br>prefetch requests monitor element . . . . .   | 1087 | pool_write_time - Total buffer pool physical write<br>time monitor element . . . . .   | 1128 |

|   |  |      |
|---|--|------|
| pool_xda_gbp_indep_pages                              | progress_start_time - Progress Start Time . . . . .  | 1161 |
| _found_in_lbp - Group buffer pool XDA                 | progress_total_units - Total Progress Work Units     | 1161 |
| independent pages found in local buffer pool          | progress_work_metric - Progress Work Metric          | 1161 |
| monitor element . . . . .                             | pseudo_deletes - Pseudo deletes monitor element      | 1162 |
| 1130  | pseudo_empty_pages - Pseudo empty pages              |      |
| pool_xda_gbp_invalid_pages - Group buffer pool        | monitor element . . . . .                            | 1162 |
| invalid XDA data pages monitor element. . . . .       | query_actual_degree - Actual runtime degree of       |      |
| 1131  | intrapartition parallelism monitor element . . . . . | 1162 |
| pool_xda_gbp_l_reads - Group buffer pool XDA          | query_card_estimate - Query Number of Rows           |      |
| data logical read requests monitor element . . . . .  | Estimate. . . . .                                    | 1163 |
| 1132  | query_cost_estimate - Query cost estimate monitor    |      |
| pool_xda_gbp_p_reads - Group buffer pool XDA          | element . . . . .                                    | 1163 |
| data physical read requests monitor element . . . . . | query_data_tag_list - Estimated query data tag list  |      |
| 1134  | monitor element . . . . .                            | 1164 |
| pool_xda_l_reads - Buffer pool XDA data logical       | queue_assignments_total - Queue assignments          |      |
| reads monitor element . . . . .                       | total monitor element . . . . .                      | 1165 |
| 1136  | queue_start_time - Queue start timestamp monitor     |      |
| pool_xda_lbp_pages_found - Local buffer pool          | element . . . . .                                    | 1165 |
| XDA data pages found monitor element . . . . .        | queue_size_top - Queue size top monitor element      | 1165 |
| 1138  | queue_time_total - Queue time total monitor          |      |
| pool_xda_p_reads - Buffer pool XDA data               | element . . . . .                                    | 1166 |
| physical reads monitor element . . . . .              | queued_agents - Queued threshold agents              |      |
| 1139  | monitor element . . . . .                            | 1166 |
| pool_xda_writes - Buffer pool XDA data writes         | quiescer_agent_id - Quiescer Agent Identification    | 1166 |
| monitor element . . . . .                             | quiescer_auth_id - Quiescer User Authorization       |      |
| 1141  | Identification . . . . .                             | 1167 |
| port_number - Port number monitor element . . . . .   | quiescer_obj_id - Quiescer Object Identification     | 1167 |
| 1143  | quiescer_state - Quiescer State . . . . .            | 1167 |
| post_shrthreshold_hash_joins - Post threshold         | quiescer_ts_id - Quiescer Table Space                |      |
| hash joins . . . . .                                  | Identification . . . . .                             | 1168 |
| 1143  | range_adjustment - Range Adjustment. . . . .         | 1168 |
| post_shrthreshold_sorts - Post shared threshold       | range_container_id - Range Container . . . . .       | 1168 |
| sorts monitor element . . . . .                       | range_end_stripe - End Stripe . . . . .              | 1168 |
| 1144  | range_max_extent - Maximum Extent in Range           | 1169 |
| post_threshold_hash_joins - Hash Join Threshold       | range_max_page_number - Maximum Page in              |      |
| 1145  | Range . . . . .                                      | 1169 |
| post_threshold_olap_funcs - OLAP Function             | range_num_containers - Number of Containers in       |      |
| Threshold monitor element . . . . .                   | Range . . . . .                                      | 1169 |
| 1146  | range_number - Range Number . . . . .                | 1169 |
| post_threshold_peas - Partial early aggregation       | range_offset - Range Offset . . . . .                | 1169 |
| threshold monitor element. . . . .                    | range_start_stripe - Start Stripe . . . . .          | 1170 |
| 1146  | range_stripe_set_number - Stripe Set Number          | 1170 |
| post_threshold_peds - Partial early distincts         | reclaim_wait_time - Reclaim wait time monitor        |      |
| threshold monitor element. . . . .                    | element . . . . .                                    | 1170 |
| 1148  | reclaimable_space_enabled - Reclaimable space        |      |
| post_threshold_sorts - Post threshold sorts           | enabled indicator monitor element . . . . .          | 1171 |
| monitor element . . . . .                             | regvar_collection_type - Registry variable           |      |
| 1150  | collection type. . . . .                             | 1172 |
| prefetch_wait_time - Time waited for prefetch         | regvar_level - Registry variable level . . . . .     | 1172 |
| monitor element . . . . .                             | regvar_name - Registry variable name. . . . .        | 1172 |
| 1151  | regvar_old_value - Registry variable old value       | 1173 |
| prefetch_waits - Prefetcher wait count monitor        | regvar_value - Registry variable value. . . . .      | 1173 |
| element . . . . .                                     | rej_curs_blk - Rejected Block Cursor Requests        | 1173 |
| 1153  | rem_cons_in - Remote Connections To Database         |      |
| prep_time - Preparation time monitor element          | Manager. . . . .                                     | 1174 |
| 1154  | rem_cons_in_exec - Remote Connections                |      |
| prep_time_best - Statement best preparation time      | Executing in the Database Manager . . . . .          | 1174 |
| monitor element . . . . .                             | remote_lock_time - Remote Lock Time . . . . .        | 1175 |
| 1155  | remote_locks - Remote Locks. . . . .                 | 1175 |
| prep_time_worst - Statement worst preparation         | remote_member - Remote member monitor                |      |
| time monitor element . . . . .                        | element . . . . .                                    | 1175 |
| 1155  |  |      |
| prev_uow_stop_time - Previous Unit of Work            |  |      |
| Completion Timestamp. . . . .                         |  |      |
| 1155  |  |      |
| priority - Priority value monitor element . . . . .   |  |      |
| 1156  |  |      |
| priv_workspace_num_overflows - Private                |  |      |
| Workspace Overflows . . . . .                         |  |      |
| 1156  |  |      |
| priv_workspace_section_inserts - Private              |  |      |
| Workspace Section Inserts . . . . .                   |  |      |
| 1157  |  |      |
| priv_workspace_section_lookups - Private              |  |      |
| Workspace Section Lookups . . . . .                   |  |      |
| 1158  |  |      |
| priv_workspace_size_top - Maximum Private             |  |      |
| Workspace Size . . . . .                              |  |      |
| 1158  |  |      |
| product_name - Product Name . . . . .                 |  |      |
| 1159  |  |      |
| progress_completed_units - Completed Progress         |  |      |
| Work Units. . . . .                                   |  |      |
| 1159  |  |      |
| progress_description - Progress Description . . . . . |  |      |
| 1160  |  |      |
| progress_list_attr - Current Progress List            |  |      |
| Attributes . . . . .                                  |  |      |
| 1160  |  |      |
| progress_list_cur_seq_num - Current Progress List     |  |      |
| Sequence Number . . . . .                             |  |      |
| 1160  |  |      |
| progress_seq_num - Progress Sequence Number           |  |      |
| 1160  |  |      |

|   |      |   |      |
|---|------|---|------|
| reopt - Reopt bind option monitor element . . . . .   | 1176 | rows_returned_top - Actual rows returned top<br>monitor element . . . . .   | 1194 |
| reorg_completion - Reorganization Completion<br>Flag . . . . .  | 1176 | rows_selected - Rows Selected . . . . .   | 1195 |
| reorg_current_counter - Reorganize Progress . . . . .   | 1176 | rows_updated - Rows updated monitor element   | 1196 |
| reorg_end - Table Reorganize End Time . . . . .   | 1177 | rows_written - Rows Written . . . . .   | 1196 |
| reorg_index_id - Index Used to Reorganize the<br>Table . . . . .  | 1177 | rqsts_completed_total - Total requests completed<br>monitor element . . . . .   | 1197 |
| reorg_long_tbspc_id - Table Space Where Long<br>Objects are Reorganized monitor element. . . . .                  | 1177 | savepoint_id - Savepoint ID . . . . .   | 1198 |
| reorg_max_counter - Total Amount of<br>Reorganization . . . . .   | 1177 | sc_work_action_set_id - Service class work action<br>set ID monitor element. . . . .                                  | 1198 |
| reorg_max_phase - Maximum Reorganize Phase  | 1178 | sc_work_class_id - Service class work class ID<br>monitor element . . . . .   | 1199 |
| reorg_phase - Table reorganization phase monitor<br>element . . . . .   | 1178 | sec_log_used_top - Maximum Secondary Log<br>Space Used. . . . .   | 1199 |
| reorg_phase_start - Reorganize Phase Start Time   | 1179 | sec_logs_allocated - Secondary Logs Allocated<br>Currently . . . . .  | 1200 |
| reorg_rows_compressed - Rows Compressed . . . . .   | 1179 | section_actuals - Section actuals monitor element   | 1200 |
| reorg_rows_rejected_for_compression - Rows<br>Rejected for Compression . . . . .                                  | 1179 | section_env - Section environment monitor<br>element . . . . .  | 1201 |
| reorg_start - Table Reorganize Start Time . . . . .   | 1179 | section_number - Section number monitor<br>element . . . . .  | 1201 |
| reorg_status - Table Reorganize Status. . . . .   | 1180 | section_type - Section type indicator monitor<br>element . . . . .  | 1202 |
| reorg_tbspc_id - Table Space Where Table or Data<br>partition is Reorganized . . . . .                            | 1180 | select_sql_stmts - Select SQL Statements Executed   | 1203 |
| reorg_type - Table Reorganize Attributes . . . . .  | 1180 | select_time - Query Response Time. . . . .  | 1203 |
| reorg_xml_regions_compressed - XML regions<br>compressed monitor element. . . . .                                 | 1181 | sequence_no - Sequence number monitor element   | 1204 |
| reorg_xml_regions_rejected_for_compression -<br>XML regions rejected for compression monitor<br>element . . . . . | 1181 | sequence_no_holding_lk - Sequence Number<br>Holding Lock . . . . .  | 1204 |
| req_agent_tid - Thread identifier for agent waiting<br>to acquire lock monitor element. . . . .                   | 1182 | server_db2_type - Database Manager Type at<br>Monitored (Server) Node . . . . .                                       | 1205 |
| req_application_handle - Identifier for application<br>waiting to acquire lock monitor element . . . . .          | 1182 | server_instance_name - Server Instance Name   | 1205 |
| req_executable_id - Identifier for statement section<br>waiting to acquire lock monitor element . . . . .         | 1182 | server_platform - Server Operating System . . . . .   | 1206 |
| req_member - Member of application waiting to<br>acquire lock monitor element. . . . .                            | 1182 | server_prdid - Server Product/Version ID . . . . .  | 1206 |
| request_exec_time_avg - Request execution time<br>average monitor element . . . . .                               | 1183 | server_version - Server Version . . . . .   | 1207 |
| rf_log_num - Log being rolled forward monitor<br>element . . . . .  | 1183 | service_class_id - Service class ID monitor<br>element . . . . .  | 1207 |
| rf_status - Log Phase . . . . .   | 1184 | service_level - Service Level . . . . .   | 1208 |
| rf_timestamp - Rollforward Timestamp . . . . .  | 1184 | service_subclass_name - Service subclass name<br>monitor element . . . . .  | 1208 |
| rf_type - Rollforward Type . . . . .  | 1184 | service_superclass_name - Service superclass<br>name monitor element . . . . .  | 1209 |
| rollback_sql_stmts - Rollback Statements<br>Attempted . . . . .   | 1184 | session_auth_id - Session authorization ID<br>monitor element . . . . .   | 1211 |
| rolled_back_agent_id - Rolled Back Agent . . . . .  | 1185 | shr_workspace_num_overflows - Shared<br>Workspace Overflows . . . . .   | 1211 |
| rolled_back_appl_id - Rolled Back Application   | 1186 | shr_workspace_section_inserts - Shared<br>Workspace Section Inserts . . . . .   | 1212 |
| rolled_back_participant_no - Rolled back<br>application participant monitor element . . . . .                     | 1186 | shr_workspace_section_lookups - Shared<br>Workspace Section Lookups . . . . .   | 1213 |
| rolled_back_sequence_no - Rolled Back Sequence<br>Number. . . . .   | 1186 | shr_workspace_size_top - Maximum Shared<br>Workspace Size . . . . .   | 1213 |
| root_node_splits - Root node splits monitor<br>element . . . . .  | 1187 | skipped_prefetch_data_p_reads - Skipped prefetch<br>data physical reads monitor element . . . . .                     | 1214 |
| routine_id - Routine ID monitor element . . . . .   | 1187 | skipped_prefetch_index_p_reads - Skipped<br>prefetch index physical reads monitor element . . . . .                   | 1215 |
| rows_deleted - Rows deleted monitor element   | 1188 | skipped_prefetch_temp_data_p_reads - Skipped<br>prefetch temporary data physical reads monitor<br>element . . . . .   | 1216 |
| rows_fetched - Rows fetched monitor element   | 1188 | skipped_prefetch_temp_index_p_reads - Skipped<br>prefetch temporary index physical reads monitor<br>element . . . . . | 1217 |
| rows_inserted - Rows inserted monitor element   | 1189 |   |      |
| rows_modified - Rows modified monitor element   | 1190 |   |      |
| rows_read - Rows read monitor element . . . . .   | 1191 |   |      |
| rows_returned - Rows returned monitor element   | 1193 |   |      |

|   |      |   |      |
|---|------|---|------|
| skipped_prefetch_temp_xda_p_reads - Skipped prefetch temporary XDA data physical reads monitor element . . . . .                  | 1218 | sqltempespace_threshold_id - SQL temporary space threshold ID monitor element . . . . .                               | 1233 |
| skipped_prefetch_uow_data_p_reads - Skipped prefetch unit of work data physical reads monitor element . . . . .                   | 1219 | sqltempespace_threshold_value - SQL temporary space threshold value monitor element . . . . .                         | 1233 |
| skipped_prefetch_uow_index_p_reads - Skipped prefetch unit of work index physical reads monitor element . . . . .                 | 1219 | sqltempespace_threshold_violated - SQL temporary space threshold violated monitor element . . . . .                   | 1233 |
| skipped_prefetch_uow_temp_data_p_reads - Skipped prefetch unit of work temporary data physical reads monitor element . . . . .    | 1220 | spacemappage_page_reclaims_x - Space map page reclaims exclusive access monitor element . . . . .                     | 1234 |
| skipped_prefetch_uow_temp_index_p_reads - Skipped prefetch unit of work temporary index physical reads monitor element . . . . .  | 1221 | spacemappage_page_reclaims_s - Space map page reclaims shared access monitor element . . . . .                        | 1234 |
| skipped_prefetch_uow_temp_xda_p_reads - Skipped prefetch unit of work temporary XDA data physical reads monitor element . . . . . | 1221 | spacemappage_page_reclaims_initiated_x - Space map page reclaims initiated exclusive access monitor element . . . . . | 1235 |
| skipped_prefetch_uow_xda_p_reads - Skipped prefetch unit of work XDA data physical reads monitor element . . . . .                | 1221 | spacemappage_page_reclaims_initiated_s - Space map page reclaims initiated shared access monitor element . . . . .    | 1235 |
| skipped_prefetch_xda_p_reads - Skipped prefetch XDA physical reads monitor element . . . . .                                      | 1222 | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element . . . . .                           | 1236 |
| smallest_log_avail_node - Node with Least Available Log Space. . . . .  | 1223 | ss_exec_time - Subsection Execution Elapsed Time  | 1237 |
| sort_heap_allocated - Total Sort Heap Allocated   | 1223 | ss_node_number - Subsection Node Number . . . . .   | 1238 |
| sort_heap_top - Sort private heap high watermark  | 1224 | ss_number - Subsection number monitor element   | 1238 |
| sort_overflows - Sort overflows monitor element   | 1224 | ss_status - Subsection status monitor element   | 1238 |
| sort_shrheap_allocated - Sort Share Heap Currently Allocated . . . . .  | 1226 | ss_sys_cpu_time - System CPU Time used by Subsection . . . . .  | 1239 |
| sort_shrheap_top - Sort share heap high watermark . . . . .   | 1226 | ss_usr_cpu_time - User CPU Time used by Subsection . . . . .  | 1239 |
| source_service_class_id - Source service class ID monitor element . . . . .   | 1227 | ssl_port_number - SSL port number monitor element . . . . .   | 1240 |
| sp_rows_selected - Rows Returned by Stored Procedures . . . . .   | 1227 | start_event_id - Start event ID . . . . .   | 1240 |
| sql_chains - Number of SQL Chains Attempted   | 1228 | start_event_timestamp - Start event timestamp   | 1240 |
| sql_req_id - Request Identifier for SQL Statement   | 1228 | start_time - Event Start Time . . . . .   | 1241 |
| sql_reqs_since_commit - SQL Requests Since Last Commit . . . . .  | 1228 | static_sql_stmts - Static SQL Statements Attempted . . . . .  | 1241 |
| sql_stmts - Number of SQL Statements Attempted  | 1229 | statistics_timestamp - Statistics timestamp monitor element . . . . .   | 1241 |
| sqlca - SQL Communications Area (SQLCA). . . . .  | 1229 | stats_cache_size - Size of statistics cache monitor element . . . . .   | 1242 |
| sqlrowsread_threshold_id - SQL rows read threshold ID monitor element . . . . .   | 1230 | stats_fabricate_time - Total time spent on statistics fabrication activities monitor element . . . . .                | 1243 |
| sqlrowsread_threshold_value - SQL rows read threshold value monitor element . . . . .   | 1230 | stats_fabrications - Total number of statistics fabrications monitor elements . . . . .                               | 1244 |
| sqlrowsread_threshold_violated - SQL rows read threshold violated monitor element. . . . .  | 1230 | status_change_time - Application Status Change Time . . . . .   | 1244 |
| sqlrowsreadinsc_threshold_id - SQL rows read in service class threshold ID monitor element . . . . .                              | 1231 | stmt_elapsed_time - Most Recent Statement Elapsed Time . . . . .  | 1245 |
| sqlrowsreadinsc_threshold_value - SQL rows read in service class threshold value monitor element . . . . .                        | 1231 | stmt_exec_time - Statement execution time monitor element . . . . .   | 1245 |
| sqlrowsreadinsc_threshold_violated - SQL rows read in service class threshold violated monitor element . . . . .                  | 1231 | stmt_first_use_time - Statement first use timestamp monitor element . . . . .   | 1246 |
| sqlrowsreturned_threshold_id - SQL rows read returned threshold ID monitor element . . . . .                                      | 1232 | stmt_history_id - Statement history identifier . . . . .  | 1246 |
| sqlrowsreturned_threshold_value - SQL rows read returned threshold value monitor element . . . . .                                | 1232 | inact_stmthist_sz - Statement history list size . . . . .   | 1247 |
| sqlrowsreturned_threshold_violated - SQL rows read returned threshold violated monitor element. . . . .                           | 1232 | stmt_invocation_id - Statement invocation identifier monitor element. . . . .   | 1247 |
|   |      | stmt_isolation - Statement isolation. . . . .   | 1248 |
|   |      | stmt_last_use_time - Statement last use timestamp monitor element . . . . .   | 1248 |
|   |      | stmt_lock_timeout - Statement lock timeout monitor element . . . . .  | 1249 |
|   |      | stmt_nest_level - Statement nesting level monitor element . . . . .   | 1249 |
|   |      | stmt_node_number - Statement Node . . . . .   | 1250 |

|  |      |   |      |
|--|------|---|------|
| stmt_operation/operation - Statement operation monitor element . . . . .                           | 1250 | tablespace_cur_pool_id - Buffer pool currently being used monitor element . . . . .                           | 1274 |
| stmt_pkgcache_id - Statement package cache identifier monitor element. . . . .                     | 1251 | tablespace_current_size - Current table space size monitor element . . . . .                                  | 1275 |
| stmt_query_id - Statement query identifier monitor element . . . . .                               | 1252 | tablespace_extent_size - Table space extent size monitor element . . . . .                                    | 1275 |
| stmt_sorts - Statement Sorts . . . . .   | 1253 | tablespace_free_pages - Free pages in table space monitor element . . . . .                                   | 1275 |
| stmt_source_id - Statement source identifier. . . . .  | 1253 | tablespace_id - Table space identification monitor element . . . . .  | 1276 |
| stmt_start - Statement Operation Start Timestamp   | 1254 | tablespace_increase_size - Increase size in bytes   | 1277 |
| stmt_stop - Statement Operation Stop Timestamp   | 1254 | tablespace_increase_size_percent - Increase size by percent monitor element . . . . .                         | 1277 |
| stmt_sys_cpu_time - System CPU Time used by Statement . . . . .                                    | 1255 | tablespace_initial_size - Initial table space size  | 1277 |
| stmt_text - SQL statement text monitor element   | 1255 | tablespace_last_resize_failed - Last resize attempt failed. . . . .   | 1278 |
| stmt_type - Statement type monitor element. . . . .  | 1256 | tablespace_last_resize_time - Time of last successful resize . . . . .  | 1278 |
| stmt_type_id - Statement type identifier monitor element . . . . .                                 | 1257 | tablespace_max_size - Maximum table space size  | 1278 |
| stmt_unicode - Statement unicode flag monitor element . . . . .                                    | 1258 | tablespace_min_recovery_time - Minimum recovery time for rollforward monitor element . . . . .                | 1279 |
| stmt_usr_cpu_time - User CPU Time used by Statement . . . . .                                      | 1258 | tablespace_name - Table space name monitor element . . . . .  | 1279 |
| stmt_value_data - Value data. . . . .  | 1259 | tablespace_next_pool_id - Buffer pool that will be used at next startup monitor element . . . . .             | 1280 |
| stmt_value_index - Value index . . . . .   | 1259 | tablespace_num_containers - Number of Containers in Table Space . . . . .                                     | 1281 |
| stmt_value_isnull - Value has null value monitor element . . . . .                                 | 1260 | tablespace_num_quiescers - Number of Quiescers  | 1281 |
| stmt_value_isreopt - Variable used for statement reoptimization monitor element . . . . .          | 1261 | tablespace_num_ranges - Number of Ranges in the Table Space Map . . . . .                                     | 1281 |
| stmt_value_type - Value type monitor element   | 1261 | tablespace_page_size - Table space page size monitor element . . . . .  | 1281 |
| sto_path_free_sz - Automatic storage path free space monitor element . . . . .                     | 1262 | tablespace_page_top - Table space high watermark monitor element . . . . .                                    | 1282 |
| stop_time - Event Stop Time . . . . .  | 1262 | tablespace_paths_dropped - Table space using dropped path monitor element . . . . .                           | 1282 |
| storage_group_id - Storage group identifier . . . . .  | 1263 | tablespace_pending_free_pages - Pending free pages in table space monitor element . . . . .                   | 1283 |
| storage_group_name - Storage group name . . . . .  | 1263 | tablespace_prefetch_size - Table space prefetch size monitor element . . . . .                                | 1283 |
| stored_proc_time - Stored Procedure Time . . . . .   | 1263 | tablespace_rebalancer_extents_processed - Number of extents the rebalancer has processed . . . . .            | 1284 |
| stored_procs - Stored Procedures . . . . .   | 1264 | tablespace_rebalancer_extents_remaining - Total number of extents to be processed by the rebalancer . . . . . | 1284 |
| swap_pages_in - Pages swapped in from disk monitor element . . . . .                               | 1264 | tablespace_rebalancer_last_extent_moved - Last extent moved by the rebalancer . . . . .                       | 1285 |
| swap_pages_out - Pages swapped out to disk monitor element . . . . .                               | 1264 | tablespace_rebalancer_mode - Rebalancer mode monitor element . . . . .  | 1285 |
| swap_page_size - Swap page size monitor element . . . . .  | 1265 | tablespace_rebalancer_priority - Current rebalancer priority . . . . .  | 1286 |
| sync_runstats - Total number of synchronous RUNSTATS activities monitor element. . . . .           | 1265 | tablespace_rebalancer_restart_time - Rebalancer restart time. . . . .   | 1287 |
| sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element . . . . . | 1266 | tablespace_rebalancer_source_storage_group_id - Rebalancer source storage group identifier . . . . .          | 1287 |
| system_auth_id - System authorization identifier monitor element . . . . .                         | 1267 | tablespace_rebalancer_source_storage_group_name - Rebalancer source storage group name . . . . .              | 1287 |
| system_cpu_time - System CPU time monitor element . . . . .  | 1267 | tablespace_rebalancer_start_time - Rebalancer start time . . . . .  | 1288 |
| tab_file_id - Table file ID monitor element . . . . .  | 1268 | tablespace_rebalancer_status - Rebalancer status monitor element . . . . .                                    | 1288 |
| tab_type - Table type monitor element. . . . .   | 1268 |   |      |
| table_file_id - Table file ID monitor element. . . . .   | 1268 |   |      |
| table_name - Table name monitor element . . . . .  | 1269 |   |      |
| table_scans - Table scans monitor element . . . . .  | 1271 |   |      |
| table_schema - Table schema name monitor element . . . . .   | 1271 |   |      |
| table_type - Table type monitor element . . . . .  | 1273 |   |      |
| tablespace_auto_resize_enabled - Table space automatic resizing enabled monitor element. . . . .   | 1273 |   |      |
| tablespace_content_type - Table space content type monitor element . . . . .                       | 1274 |   |      |

|  |      |   |      |
|--|------|---|------|
| tablespace_rebalancer_target_storage_group_id -<br>Rebalancer target storage group identifier . . . . .  | 1288 | threshold_queuesize - Threshold queue size<br>monitor element . . . . .   | 1308 |
| tablespace_rebalancer_target_storage_group_name<br>- Rebalancer target storage group name . . . . .      | 1289 | thresholdid - Threshold ID monitor element. . . . .   | 1308 |
| tablespace_state - Table space state monitor<br>element . . . . .  | 1289 | time_completed - Time completed monitor<br>element . . . . .  | 1308 |
| tablespace_state_change_object_id - State Change<br>Object Identification. . . . .                       | 1291 | time_created - Time created monitor element . . . . .   | 1309 |
| tablespace_state_change_ts_id - State Change<br>Table Space Identification . . . . .                     | 1291 | time_of_violation - Time of violation monitor<br>element . . . . .  | 1309 |
| tablespace_total_pages - Total pages in table space<br>monitor element . . . . .                         | 1292 | time_stamp - Snapshot Time . . . . .  | 1309 |
| tablespace_type - Table space type monitor<br>element . . . . .  | 1292 | time_started - Time started monitor element. . . . .  | 1310 |
| tablespace_usable_pages - Usable pages in table<br>space monitor element . . . . .                       | 1293 | time_zone_disp - Time Zone Displacement . . . . .   | 1310 |
| tablespace_used_pages - Used pages in table<br>space monitor element . . . . .                           | 1293 | top - Histogram bin top monitor element. . . . .  | 1310 |
| tablespace_using_auto_storage - Table space<br>enabled for automatic storage monitor element. . . . .    | 1294 | tot_log_used_top - Maximum Total Log Space<br>Used . . . . .  | 1310 |
| target_cf_gbp_size - Target cluster caching facility<br>group buffer pool size monitor element . . . . . | 1294 | total_act_time - Total activity time monitor<br>element . . . . .   | 1311 |
| target_cf_lock_size - Target cluster caching facility<br>lock size monitor element . . . . .             | 1295 | total_act_wait_time - Total activity wait time<br>monitor element . . . . .   | 1312 |
| target_cf_sca_size - Target cluster caching facility<br>shared communications area size monitor element  | 1295 | total_app_commits - Total application commits<br>monitor elements. . . . .  | 1314 |
| tbasp_datatag - Table space data tag. . . . .  | 1295 | total_app_rollback - Total application rollbacks<br>monitor element . . . . .   | 1315 |
| tbasp_last_consec_page - Last consecutive object<br>table page monitor element . . . . .                 | 1296 | total_app_rqst_time - Total application request<br>time monitor element . . . . .   | 1316 |
| tbasp_max_page_top - Maximum table space page<br>high watermark monitor element . . . . .                | 1296 | total_app_section_executions - Total application<br>section executions monitor element. . . . .                                 | 1317 |
| tbasp_names - Table space names . . . . .  | 1296 | total_buffers_rcvd - Total FCM Buffers Received   | 1318 |
| tbasp_trackmod_state - Table space trackmod state<br>monitor element . . . . .                           | 1296 | total_buffers_sent - Total FCM Buffers Sent . . . . .   | 1318 |
| tcpip_recv_volume - TCP/IP received volume<br>monitor element . . . . .                                  | 1297 | total_bytes_received - Bytes received monitor<br>element . . . . .  | 1319 |
| tcpip_recv_wait_time - TCP/IP received wait time<br>monitor element . . . . .                            | 1298 | total_bytes_sent - Bytes sent monitor element   | 1319 |
| tcpip_recvs_total - TCP/IP receives total monitor<br>element . . . . .                                   | 1299 | total_commit_proc_time - Total commits<br>processing time monitor element . . . . .   | 1319 |
| tcpip_send_volume - TCP/IP send volume<br>monitor element . . . . .                                      | 1300 | total_commit_time - Total commit time monitor<br>element . . . . .  | 1320 |
| tcpip_send_wait_time - TCP/IP send wait time<br>monitor element . . . . .                                | 1301 | total_compilations - Total compilations monitor<br>element . . . . .  | 1321 |
| tcpip_sends_total - TCP/IP sends total monitor<br>element . . . . .                                      | 1302 | total_compile_proc_time - Total compile<br>processing time monitor element . . . . .  | 1322 |
| temp_tablespace_top - Temporary table space top<br>monitor element . . . . .                             | 1303 | total_compile_time - Total compile time monitor<br>element . . . . .  | 1323 |
| territory_code - Database Territory Code . . . . .   | 1303 | total_cons - Connects Since Database Activation   | 1324 |
| thresh_violations - Number of threshold violations<br>monitor element . . . . .                          | 1304 | total_connect_authentication_proc_time - Total<br>connection authentication processing time monitor<br>element . . . . .        | 1325 |
| threshold_action - Threshold action monitor<br>element . . . . .   | 1305 | total_connect_authentications - Connections or<br>switch user authentications performed monitor<br>element . . . . .            | 1326 |
| threshold_domain - Threshold domain monitor<br>element . . . . .   | 1305 | total_connect_authentication_time - Total<br>connection or switch user authentication request<br>time monitor element . . . . . | 1327 |
| threshold_maxvalue - Threshold maximum value<br>monitor element . . . . .                                | 1306 | total_connect_request_proc_time - Total<br>connection or switch user request processing time<br>monitor element . . . . .       | 1328 |
| threshold_name - Threshold name monitor<br>element . . . . .   | 1306 | total_connect_requests - Connection or switch<br>user requests monitor element . . . . .  | 1329 |
| threshold_predicate - Threshold predicate monitor<br>element . . . . .                                   | 1307 | total_connect_request_time - Total connection or<br>switch user request time monitor element . . . . .                          | 1330 |
|  |      | total_cpu_time - Total CPU time monitor element   | 1330 |
|  |      | total_disp_run_queue_time - Total dispatcher run<br>queue time monitor element . . . . .  | 1332 |

|  |      |  |      |
|--|------|--|------|
| total_exec_time - Elapsed statement execution<br>time monitor element . . . . .                          | 1334 | total_runstats_proc_time - Total runtime statistics<br>processing time monitor element . . . . .             | 1364 |
| total_extended_latch_wait_time - Total extended<br>latch wait time monitor element. . . . .              | 1335 | total_runstats_time - Total runtime statistics time<br>monitor element . . . . .                             | 1365 |
| total_extended_latch_waits - Total extended latch<br>waits monitor element . . . . .                     | 1336 | total_sec_cons - Secondary Connections . . . . .   | 1366 |
| total_move_time - Total extent move time monitor<br>element . . . . .                                    | 1338 | total_section_proc_time - Total section processing<br>time monitor element . . . . .                         | 1367 |
| total_hash_joins - Total Hash Joins . . . . .  | 1338 | total_section_sort_proc_time - Total section sort<br>processing time monitor element . . . . .               | 1368 |
| total_hash_loops - Total Hash Loops . . . . .  | 1338 | total_section_sort_time - Total section sort time<br>monitor element . . . . .                               | 1370 |
| total_implicit_compilations - Total implicit<br>compilations monitor element . . . . .                   | 1339 | total_section_sorts - Total section sorts monitor<br>element . . . . .                                       | 1371 |
| total_implicit_compile_proc_time - Total implicit<br>compile processing time monitor element . . . . .   | 1340 | total_section_time - Total section time monitor<br>element . . . . .   | 1373 |
| total_implicit_compile_time - Total implicit<br>compile time monitor element . . . . .                   | 1341 | total_sort_time - Total sort time monitor element  | 1374 |
| total_load_proc_time - Total load processing time<br>monitor element . . . . .                           | 1342 | total_sorts - Total sorts monitor element . . . . .  | 1375 |
| total_load_time - Total load time monitor element  | 1343 | total_stats_fabrication_proc_time - Total statistics<br>fabrication processing time monitor element. . . . . | 1377 |
| total_loads - Total loads monitor element. . . . .   | 1344 | total_stats_fabrication_time - Total statistics<br>fabrication time monitor element . . . . .                | 1378 |
| total_log_available - Total Log Available . . . . .  | 1345 | total_stats_fabrications - Total statistics<br>fabrications monitor elements . . . . .                       | 1379 |
| total_log_used - Total Log Space Used. . . . .   | 1345 | total_sync_runstats_time - Total synchronous<br>RUNSTATS time monitor elements . . . . .                     | 1380 |
| total_move_time - Total extent move time monitor<br>element . . . . .                                    | 1346 | total_sync_runstats_proc_time - Total synchronous<br>RUNSTATS processing time monitor element . . . . .      | 1381 |
| total_olap_funcs - Total OLAP Functions monitor<br>element . . . . .                                     | 1346 | total_sync_runstats - Total synchronous<br>RUNSTATS activities monitor element. . . . .                      | 1382 |
| total_peas - Total partial early aggregations<br>monitor element . . . . .                               | 1346 | total_sys_cpu_time - Total system CPU time for a<br>statement monitor element . . . . .                      | 1383 |
| total_peds - Total partial early distincts monitor<br>element . . . . .                                  | 1349 | total_sorts - Total sorts monitor element . . . . .  | 1384 |
| total_reorg_proc_time - Total reorganization<br>processing time monitor element . . . . .                | 1350 | total_usr_cpu_time - Total user CPU time for a<br>statement monitor element . . . . .                        | 1385 |
| total_reorg_time - Total reorganization time<br>monitor element . . . . .                                | 1351 | total_wait_time - Total wait time monitor element  | 1386 |
| total_reorgs - Total reorganizations monitor<br>element . . . . .  | 1352 | tpmon_acc_str - TP monitor client accounting<br>string monitor element . . . . .                             | 1387 |
| total_rollback_proc_time - Total rollback<br>processing time monitor element . . . . .                   | 1353 | tpmon_client_app - TP monitor client application<br>name monitor element . . . . .                           | 1388 |
| total_rollback_time - Total rollback time monitor<br>element . . . . .                                   | 1354 | tpmon_client_userid - TP monitor client user ID<br>monitor element . . . . .                                 | 1388 |
| total_routine_invocations - Total routine<br>invocations monitor elements . . . . .                      | 1355 | tpmon_client_wkstn - TP monitor client<br>workstation name monitor element. . . . .                          | 1389 |
| total_routine_non_sect_proc_time - Non-section<br>processing time monitor element . . . . .              | 1356 | tq_cur_send_spills - Current number of table<br>queue buffers overflowed monitor element . . . . .           | 1389 |
| total_routine_non_sect_time - Non-section routine<br>execution time monitor elements . . . . .           | 1356 | tq_id_waiting_on - Waited on node on a table<br>queue monitor element. . . . .                               | 1390 |
| total_routine_time - Total routine time monitor<br>element . . . . .                                     | 1357 | tq_max_send_spills - Maximum number of table<br>queue buffers overflows . . . . .                            | 1390 |
| total_routine_user_code_proc_time - Total routine<br>user code processing time monitor element . . . . . | 1359 | tq_node_waited_for - Waited for node on a table<br>queue . . . . .   | 1391 |
| total_routine_user_code_time - Total routine user<br>code time monitor element . . . . .                 | 1360 | tq_rows_read - Number of Rows Read from table<br>queues . . . . .  | 1391 |
| total_rqst_mapped_in - Total request mapped-in<br>monitor element . . . . .                              | 1362 | tq_rows_written - Number of rows written to<br>table queues . . . . .  | 1391 |
| total_rqst_mapped_out - Total request<br>mapped-out monitor element. . . . .                             | 1362 | tq_sort_heap_rejections - Table queue sort heap<br>rejections monitor element . . . . .                      | 1392 |
| total_rqst_time - Total request time monitor<br>element . . . . .  | 1362 | tq_sort_heap_requests - Table queue sort heap<br>requests monitor element . . . . .                          | 1394 |
| total_runstats - Total runtime statistics monitor<br>element . . . . .                                   | 1363 | tq_tot_send_spills - Total number of table queue<br>buffers overflowed monitor element . . . . .             | 1396 |

|   |      |
|---|------|
| tq_wait_for_any - Waiting for any node to send on a table queue . . . . .                     | 1397 |
| ts_name - Table space being rolled forward monitor element . . . . .                          | 1397 |
| txn_completion_status - Transaction completion status . . . . .                               | 1398 |
| uid_sql_stmts - Update/Insert/Delete SQL Statements Executed. . . . .                         | 1398 |
| unread_prefetch_pages - Unread prefetch pages monitor element . . . . .                       | 1399 |
| uow_comp_status - Unit of Work Completion Status . . . . .                                    | 1399 |
| uow_completed_total - Total completed units of work monitor element . . . . .                 | 1400 |
| uow_elapsed_time - Most Recent Unit of Work Elapsed Time . . . . .                            | 1400 |
| uow_id - Unit of work ID monitor element . . . . .  | 1401 |
| uow_lifetime_avg - Unit of work lifetime average monitor element . . . . .                    | 1402 |
| uow_lock_wait_time - Total time unit of work waited on locks monitor element . . . . .        | 1403 |
| uow_log_space_used - Unit of work log space used monitor element . . . . .                    | 1403 |
| uow_start_time - Unit of work start timestamp monitor element . . . . .                       | 1404 |
| uow_status - Unit of Work Status . . . . .  | 1405 |
| uow_stop_time - Unit of work stop timestamp monitor element . . . . .                         | 1405 |
| uow_throughput - Unit of work throughput monitor element . . . . .                            | 1406 |
| uow_total_time_top - UOW total time top monitor element . . . . .                             | 1406 |
| update_sql_stmts - Updates . . . . .  | 1407 |
| update_time - Update Response Time . . . . .  | 1407 |
| usage_list_last_state_change - Last state change monitor element . . . . .                    | 1408 |
| usage_list_last_updated - Usage list last updated monitor element . . . . .                   | 1408 |
| usage_list_mem_size - Usage list memory size monitor element . . . . .                        | 1408 |
| usage_list_name - Usage list name monitor element . . . . .                                   | 1409 |
| usage_list_num_references - Number of references monitor element . . . . .                    | 1409 |
| usage_list_num_ref_with_metrics - Number of references with metrics monitor element . . . . . | 1409 |
| usage_list_schema - Usage list schema monitor element . . . . .                               | 1410 |
| usage_list_size - Usage list size monitor element . . . . .                                   | 1410 |
| usage_list_state - Usage list state monitor element . . . . .                                 | 1410 |
| usage_list_used_entries - Usage list used entries monitor element . . . . .                   | 1410 |
| usage_list_wrapped - Usage list wrap indicator monitor element . . . . .                      | 1411 |
| user_cpu_time - User CPU time monitor element . . . . .                                       | 1411 |
| utility_dbname - Database Operated on by Utility . . . . .                                    | 1411 |
| utility_description - Utility Description . . . . .   | 1412 |
| utility_detail - Utility detail . . . . .   | 1412 |
| utility_id - Utility ID . . . . .   | 1412 |
| utility_invocation_id - Utility invocation ID . . . . .                                       | 1413 |
| utility_invoker_type - Utility Invoker Type . . . . .   | 1413 |

|  |      |
|--|------|
| utility_operation_type - Utility operation type . . . . .  | 1414 |
| utility_phase_detail - Utility phase detail . . . . .  | 1415 |
| utility_phase_type - Utility phase type . . . . .  | 1415 |
| utility_priority - Utility Priority . . . . .  | 1415 |
| utility_start_time - Utility Start Time . . . . .  | 1416 |
| utility_start_type - Utility start type . . . . .  | 1416 |
| utility_state - Utility State . . . . .  | 1416 |
| utility_stop_type - Utility stop type . . . . .  | 1417 |
| valid - Section validity indicator monitor element . . . . .                                     | 1417 |
| utility_type - Utility Type . . . . .  | 1417 |
| valid - Section validity indicator monitor element . . . . .                                     | 1418 |
| vectored_ios - Number of vectored I/O requests monitor element . . . . .                         | 1419 |
| version - Version of Monitor Data . . . . .  | 1419 |
| virtual_mem_free - Free virtual memory monitor element . . . . .                                 | 1419 |
| virtual_mem_reserved - Reserved virtual memory monitor element . . . . .                         | 1420 |
| virtual_mem_total - Total virtual memory monitor element . . . . .                               | 1420 |
| wl_work_action_set_id - Workload work action set identifier monitor element . . . . .            | 1420 |
| wl_work_class_id - Workload work class identifier monitor element . . . . .                      | 1421 |
| wlm_queue_assignments_total - Workload manager total queue assignments monitor element . . . . . | 1421 |
| wlm_queue_time_total - Workload manager total queue time monitor element . . . . .               | 1422 |
| wlo_completed_total - Workload occurrences completed total monitor element . . . . .             | 1424 |
| work_action_set_id - Work action set ID monitor element . . . . .                                | 1424 |
| work_action_set_name - Work action set name monitor element . . . . .                            | 1424 |
| work_class_id - Work class ID monitor element . . . . .  | 1425 |
| work_class_name - Work class name monitor element . . . . .                                      | 1425 |
| workload_id - Workload ID monitor element . . . . .  | 1426 |
| workload_name - Workload name monitor element . . . . .  | 1427 |
| workload_occurrence_id - Workload occurrence identifier monitor element. . . . .                 | 1428 |
| workload_occurrence_state - Workload occurrence state monitor element . . . . .                  | 1428 |
| x_lock_escals - Exclusive lock escalations monitor element . . . . .                             | 1429 |
| xda_object_pages - XDA Object Pages . . . . .  | 1430 |
| xda_object_l_pages - XML storage object (XDA) data logical pages monitor element. . . . .        | 1430 |
| xid - Transaction ID . . . . .   | 1431 |
| xmlid - XML ID monitor element . . . . .   | 1431 |
| xquery_stmts - XQuery Statements Attempted . . . . .   | 1431 |

---

## Part 3. Monitoring in a DB2 pureScale environment . . . . . 1433

### Chapter 12. Status monitoring of a DB2 pureScale instance . . . . . 1435

|  |      |
|--|------|
| Interfaces for retrieving status information for DB2 pureScale instances . . . . . | 1435 |
|--|------|

|   |      |
|---|------|
| Values for member and cluster caching facility states and alerts . . . . .                                  | 1437 |
| Interpretation of status information. . . . .   | 1439 |
| Examples: Viewing the status of hosts, members and cluster caching facilities . . . . .                     | 1444 |
| Viewing status information for hosts in a DB2 pureScale instance . . . . .                                  | 1445 |
| Viewing status information for members and cluster caching facilities in a DB2 pureScale instance . . . . . | 1446 |
| Checking restart status for members . . . . .   | 1450 |
| Viewing details for an alert . . . . .  | 1452 |

### **Chapter 13. Event and real-time database and system monitoring in a DB2 pureScale environment . . . . . 1455**

|  |      |
|--|------|
| Cluster caching facility memory and CPU usage monitoring overview . . . . .                  | 1457 |
| Monitor elements for viewing cluster caching facility memory usage . . . . .                 | 1458 |
| Retrieving information from cluster caching facility memory usage monitor elements . . . . . | 1459 |
| Viewing cluster caching facility processor load . . . . .                                    | 1461 |
| Buffer pool monitoring in a DB2 pureScale environment . . . . .                              | 1463 |
| Monitor elements for viewing DB2 pureScale buffer pool activity . . . . .                    | 1463 |
| Buffer pool hit rates and hit ratios in a DB2 pureScale environment . . . . .                | 1465 |
| Lock monitoring in a DB2 pureScale environment overview . . . . .                            | 1473 |
| Lock requests between members . . . . .  | 1473 |
| Monitor elements for viewing locks between members . . . . .                                 | 1475 |
| Page reclaiming . . . . .  | 1476 |
| Monitor elements for page reclaiming . . . . .   | 1476 |
| Monitoring page reclaiming between members . . . . .   | 1477 |

### **Chapter 14. Using deprecated monitoring features in a DB2 pureScale environment. . . . . 1481**

### **Chapter 15. New and changed monitor elements . . . . . 1487**

|  |      |
|--|------|
| cf_wait_time - cluster caching facility wait time monitor element . . . . .  | 1487 |
| cf_waits - Number of cluster caching facility waits monitor element . . . . .  | 1487 |
| configured_cf_gbp_size - Configured cluster caching facility group buffer pool size monitor element . . . . .          | 1488 |
| configured_cf_lock_size - Configured cluster caching facility lock size monitor element . . . . .                      | 1488 |
| configured_cf_mem_size - Configured cluster caching facility memory size monitor element . . . . .                     | 1489 |
| configured_cf_sca_size - Configured cluster caching facility shared communications area size monitor element . . . . . | 1489 |

|   |      |
|---|------|
| current_cf_gbp_size - Current cluster caching facility group buffer pool size monitor element. . . . .            | 1489 |
| current_cf_lock_size - Current cluster caching facility lock size monitor element . . . . .                       | 1489 |
| current_cf_mem_size - Current cluster caching facility memory size monitor element . . . . .                      | 1490 |
| current_cf_sca_size - Current cluster caching facility shared communications area size monitor element . . . . .  | 1490 |
| db_name - Database name monitor element . . . . .   | 1490 |
| dbpartitionnum - Database partition number monitor element . . . . .  | 1491 |
| host_name - Host name monitor element. . . . .  | 1493 |
| id - cluster caching facility identification monitor element . . . . .  | 1493 |
| lock_escals - Number of lock escalations monitor element . . . . .  | 1494 |
| lock_escals_global - Number of global lock escalations monitor element . . . . .                                  | 1496 |
| lock_escals_locklist - Number of locklist lock escalations monitor element . . . . .                              | 1497 |
| lock_escals_maxlocks - Number of maxlocks lock escalations monitor element . . . . .                              | 1498 |
| lock_timeouts_global - Lock timeouts global monitor element . . . . .   | 1500 |
| lock_wait_time_global - Lock wait time global monitor element . . . . .   | 1501 |
| lock_wait_time_global_top - Top global lock wait time monitor element . . . . .                                   | 1502 |
| lock_waits_global - Lock waits global monitor element . . . . .   | 1503 |
| member - Database member monitor element . . . . .  | 1504 |
| objtype - Object type monitor element. . . . .  | 1508 |
| page_reclaims_initiated_s - Page reclaims initiated shared access monitor element . . . . .                       | 1509 |
| page_reclaims_initiated_x - Page reclaims initiated exclusive access monitor element . . . . .                    | 1509 |
| page_reclaims_s - Page reclaims shared access monitor element . . . . .   | 1509 |
| page_reclaims_x - Page reclaims exclusive access monitor element . . . . .  | 1510 |
| pool_async_data_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element . . . . .   | 1510 |
| pool_async_data_gbp_l_reads - Asynchronous group buffer pool data logical reads monitor element . . . . .         | 1510 |
| pool_async_data_gbp_p_reads - Asynchronous group buffer pool data physical reads monitor element . . . . .        | 1511 |
| pool_async_data_lbp_pages_found - Asynchronous local buffer pool data pages found monitor element . . . . .       | 1511 |
| pool_async_index_gbp_invalid_pages - Asynchronous group buffer pool invalid index pages monitor element . . . . . | 1512 |
| pool_async_index_gbp_l_reads - Asynchronous group buffer pool index logical reads monitor element . . . . .       | 1512 |

|   |      |
|---|------|
| pool_async_index_gbp_p_reads - Asynchronous group buffer pool index physical reads monitor element . . . . .          | 1513 |
| pool_async_index_lbp_pages_found - Asynchronous local buffer pool index pages found monitor element. . . . .          | 1513 |
| pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages monitor element . . . . .    | 1514 |
| pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests monitor element . . . . .  | 1514 |
| pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests monitor element . . . . . | 1515 |
| pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found monitor element . . . . .        | 1515 |
| pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element . . . . .                          | 1516 |
| pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element . . . . .                                | 1517 |
| pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element . . . . .                               | 1519 |
| pool_data_lbp_pages_found - Local buffer pool found data pages monitor element . . . . .                              | 1520 |
| pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element . . . . .                        | 1522 |
| pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element . . . . .                              | 1523 |
| pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements . . . . .                            | 1525 |
| pool_index_lbp_pages_found - Local buffer pool index pages found monitor element . . . . .                            | 1526 |
| pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages monitor element. . . . .                        | 1528 |
| pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests monitor element . . . . .                     | 1529 |
| pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests monitor element . . . . .                    | 1531 |
| reclaim_wait_time - Reclaim wait time monitor element . . . . .   | 1534 |
| spacemappage_page_reclaims_initiated_s - Space map page reclaims initiated shared access monitor element . . . . .    | 1535 |

|   |      |
|---|------|
| spacemappage_page_reclaims_initiated_x - Space map page reclaims initiated exclusive access monitor element . . . . . | 1536 |
| spacemappage_page_reclaims_s - Space map page reclaims shared access monitor element . . . . .                        | 1536 |
| spacemappage_page_reclaims_x - Space map page reclaims exclusive access monitor element . . . . .                     | 1537 |
| spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element . . . . .                           | 1537 |
| table_name - Table name monitor element . . . . .   | 1539 |
| table_schema - Table schema name monitor element . . . . .  | 1540 |
| tablespace_min_recovery_time - Minimum recovery time for rollforward monitor element . . . . .                        | 1542 |
| target_cf_gbp_size - Target cluster caching facility group buffer pool size monitor element . . . . .                 | 1543 |
| target_cf_lock_size - Target cluster caching facility lock size monitor element . . . . .                             | 1543 |
| target_cf_sca_size - Target cluster caching facility shared communications area size monitor element . . . . .        | 1543 |

---

## Part 4. Appendixes . . . . . 1545

### Appendix A. Overview of the DB2 technical information . . . . . 1547

|  |      |
|--|------|
| DB2 technical library in hardcopy or PDF format  | 1547 |
| Displaying SQL state help from the command line processor . . . . .                                  | 1550 |
| Accessing different versions of the DB2 Information Center . . . . .                                 | 1550 |
| Updating the DB2 Information Center installed on your computer or intranet server . . . . .          | 1550 |
| Manually updating the DB2 Information Center installed on your computer or intranet server . . . . . | 1552 |
| DB2 tutorials . . . . .  | 1553 |
| DB2 troubleshooting information . . . . .  | 1554 |
| Terms and conditions . . . . .   | 1554 |

### Appendix B. Notices . . . . . 1557

### Index . . . . . 1561

---

## About this book

The *System Monitor Guide and Reference* describes how to collect different kinds of information about your database and the database manager.

It also explains how you can use the information you collected to understand database activity, improve performance, and determine the cause of problems.



---

## Part 1. Interfaces for database monitoring

There are two ways to monitor operations in your database. You can view information that shows the state of various aspects of the database at a specific point in time. Or, you can set up event monitors to capture historical information as specific types of database events take place.

You can monitor your database operations in real-time using monitoring table functions. For example, you can use a monitoring table function to examine the total amount of space used in a table space. These table functions let you examine *monitor elements* and metrics that report on virtually all aspects of database operations using SQL. The monitoring table functions use the newer, lightweight, high-speed monitoring infrastructure that was introduced in Version 9.7. In addition to the table functions, snapshot monitoring routines are also available. The snapshot monitoring facilities in DB2® use monitoring infrastructure that existed before Version 9.7. Generally speaking, snapshot monitoring facilities are no longer being enhanced in the product; where possible, use the monitoring table functions to retrieve the data you want to see.

Event monitors capture information about database operations over time, as specific types of events occur. For example, you can create an event monitor to capture information about locks and deadlocks as they occur in the system. Or you might create an event monitor to record when a threshold that you specify (for example the total processor time used by an application or workload) is exceeded. Event monitors generate output in different formats; all of them can write event data to regular tables; some event monitors have additional output options.

IBM® InfoSphere® Optim™ Performance Manager provides a Web interface that you can use to isolate and analyze typical database performance problems. You can also view a summary of the health of your databases and drill down. For more details, see Monitoring with Optim Performance Manager at [http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p\\_monitor.html](http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p_monitor.html).



---

## Chapter 1. Database monitoring

The term *database monitoring* refers to the tasks associated with examining the operational status of your database.

Database monitoring is a vital activity for the maintenance of the performance and health of your database management system. To facilitate monitoring, DB2 collects information from the database manager, its databases, and any connected applications. With this information you can perform the following types of tasks, and more:

- Forecast hardware requirements based on database usage patterns.
- Analyze the performance of individual applications or SQL queries.
- Track the usage of indexes and tables.
- Pinpoint the cause of poor system performance.
- Assess the impact of optimization activities (for example, altering database manager configuration parameters, adding indexes, or modifying SQL queries).



---

## Chapter 2. Table functions for monitoring

Starting with DB2 Version 9.7, you can access monitor data through a light-weight alternative to the traditional system monitor. Use monitor table functions to collect and view data for systems, activities, or data objects.

Data for monitored elements are continually accumulated in memory and available for querying. You can choose to receive data for a single object (for example, service class A or table TABLE1) or for all objects.

When using these table functions in a database partitioned environment, you can choose to receive data for a single partition or for all partitions. If you choose to receive data for all partitions, the table functions return one row for each partition. Using SQL, you can sum the values across partitions to obtain the value of a monitor element across partitions.

---

### Monitoring system information using table functions

The system monitoring perspective encompasses the complete volume of work and effort expended by the data server to process application requests. From this perspective, you can determine what the data server is doing as a whole as well as for particular subsets of application requests.

Monitor elements for this perspective, referred to as request monitor elements, cover the entire range of data server operations associated with processing requests.

Request monitor elements are continually accumulated and aggregated in memory so they are immediately available for querying. Request monitor elements are aggregated across requests at various levels of the workload management (WLM) object hierarchy: by unit of work, by workload, by service class. They are also aggregated by connection.

Use the following table functions for accessing current system monitoring information:

- MON\_GET\_SERVICE\_SUBCLASS and MON\_GET\_SERVICE\_SUBCLASS\_DETAILS
- MON\_GET\_WORKLOAD and MON\_GET\_WORKLOAD\_DETAILS
- MON\_GET\_CONNECTION and MON\_GET\_CONNECTION\_DETAILS
- MON\_GET\_UNIT\_OF\_WORK and MON\_GET\_UNIT\_OF\_WORK\_DETAILS

This set of table functions enables you to drill down or focus on request monitor elements at a particular level of aggregation. Table functions are provided in pairs: one for relational access to commonly used data and the other for XML access to the complete set of available monitor elements.

The system monitoring information is collected by these table functions by default for a new database. You can change default settings using one or both of the following settings:

- The database configuration parameter **mon\_req\_metrics** specifies the minimum level of collection in all service classes.

- The COLLECT REQUEST METRICS clause of the CREATE/ALTER SERVICE CLASS statement specifies the level of collection for a service superclass. Use this setting to increase the level of collection for a given service class over the minimum level of collection set for all service classes.

The possible values for each setting are the following:

**None** No request monitor elements are collected

**Base** All request monitor elements are collected

For example, to collect system monitoring information for only a subset of service classes, do the following:

1. Set the database configuration parameter **mon\_req\_metrics** to NONE.
2. For each required service class, set the COLLECT REQUEST METRICS clause of the CREATE/ALTER SERVICE CLASS statement to BASE.

---

## Monitoring activities using table functions

The activity monitoring perspective focuses on the subset of data server processing related to executing activities. In the context of SQL statements, the term activity refers to the execution of the section for a SQL statement.

Monitor elements for this perspective, referred to as activity monitor elements, are a subset of the request monitor elements. Activity monitor elements measure aspects of work done for statement section execution. Activity monitoring includes other information such as SQL statement text for the activity.

For activities in progress, activity metrics are accumulated in memory. For activities that are SQL statements, activity metrics are also accumulated in the package cache. In the package cache activity metrics are aggregated over all executions of each SQL statement section.

Use the following table functions to access current data for activities:

### **MON\_GET\_ACTIVITY\_DETAILS**

Returns data about the individual activities in progress when the table function is called. Data is returned in a relational form, however, the detailed metrics are returned in an XML document in the DETAILS column of the results table.

### **MON\_GET\_PKG\_CACHE\_STMT**

Returns a point-in-time view of both static and dynamic SQL statements in the database package cache. Data is returned in a relational form.

### **MON\_GET\_PKG\_CACHE\_STMT\_DETAILS**

Returns detailed metrics for one or more package cache entries. Data is returned in a relational form, however, the detailed metrics are returned in an XML document in the DETAILS column of the results table.

Activity monitoring information is collected by default for a new database. You can change default settings using one or both of the following settings:

- The **mon\_act\_metrics** database configuration parameter specifies the minimum level of collection in all workloads.
- The COLLECT ACTIVITY METRICS clause of the CREATE/ALTER WORKLOAD statement specifies the level of collection for a given workload over the minimum level of collection set for all workloads.

The possible values for each setting are the following:

**None** No activity monitor elements are collected

**Base** All activity monitor elements are collected

For example, to collect activity monitor elements for only selected workloads, do the following:

1. Set the **mon\_act\_metrics** database configuration parameter to NONE.
2. Set the COLLECT ACTIVITY METRICS clause of the CREATE/ALTER WORKLOAD statement to BASE. By default, the values for other workloads is NONE.

---

## Monitoring data objects using table functions

The data object monitoring perspective provides information about operations performed on data objects, that is tables, indexes, buffer pools, table spaces, and containers.

A different set of monitor elements is available for each object type. Monitor elements for a data object are incremented each time a request involves processing that object. For example, when processing a request that involves reading rows from a particular table, the metric for rows read is incremented for that table.

Use the following table functions to access current details for data objects:

- MON\_GET\_BUFFERPOOL
- MON\_GET\_TABLESPACE
- MON\_GET\_CONTAINER
- MON\_GET\_TABLE
- MON\_GET\_INDEX

These table functions return data in a relational form.

You cannot access historical data for data objects.

Data object monitor elements are collected by default for new databases. You can use the **mon\_obj\_metrics** database configuration parameter to reduce the amount of data collected by the table functions.

The possible values for this configuration parameter are the following:

**None** No data object monitor elements are collected

**Base** Some data object monitor elements are collected

**Extended**

All data object monitor elements are collected

To stop collecting data object monitor elements reported by the following table functions, set the **mon\_obj\_metrics** configuration parameter to NONE.

- MON\_GET\_BUFFERPOOL
- MON\_GET\_TABLESPACE
- MON\_GET\_CONTAINER

## Object usage

When SQL statements are executed, they use various database objects, such as tables and indexes. Knowing which database objects a statement accesses and how the statement affects them can help you identify targets for monitoring or performance tuning.

The following table shows the entities that you can use to explore the relationship between database objects and statements.

*Table 1. Ways to identify object usage*

| Mechanism                    | Definition  | Usage   |
|------------------------------|---|---|
| Usage list                   | A usage list is a database object that records each DML statement section that references a particular table or index and captures statistics about that section as it executes.  | Identify the statements that affected a table or index. If you notice an unusual value for a metric when monitoring a database object, use a usage list to determine whether a particular statement contributed to that metric. You can also view statistics for each statement that affected the object. |
| Section explain with actuals | A section explain is a set of information about the access plan that the optimizer chose for an SQL statement. You can capture section actuals as part of the explain. Section actuals are runtime statistics that are collected when a section executes. | Identify the tables or indexes that a statement affects. You can view statistics for each table or index and use these statistics to determine how the statement affects each object and where tuning might be required.  |

You can use the information in a usage list or section explain with actuals as baseline data for performance tuning. Collect information about object usage before tuning statements or database configuration parameters. After tuning, collect the information again to verify that tuning improved performance.

### Identifying the statements that affect a table

Use usage lists to identify DML statement sections that affect a particular table when the statement sections execute. You can view statistics for each statement and use these statistics to determine where additional monitoring or tuning might be required.

### Before you begin

Do the following tasks:

- Identify a table for which you want to view object usage statistics. You can use the `MON_GET_TABLE` table function to view monitor metrics for one or more tables.
- To issue the required statements, ensure that the privileges that are held by the authorization ID of each statement include DBADM authority or SQLADM authority.
- Ensure that you have EXECUTE privilege on the `MON_GET_TABLE_USAGE_LIST` and `MON_GET_USAGE_LIST_STATUS` table functions.

## About this task

When you view the output of the `MON_GET_TABLE` table function, you might see an unusual value for a monitor element. You can use usage lists to determine whether any DML statements contributed to this value.

Usage lists contain statistics about factors like locks and buffer pool usage for each statement that affected a table during a particular time frame. If you determine that a statement affected a table negatively, use these statistics to determine where further monitoring might be required or how the statement can be tuned.

## Procedure

To identify the statements that affect a table:

1. Set the `mon_obj_metrics` configuration parameter to `EXTENDED` by issuing the following command:

```
DB2 UPDATE DATABASE CONFIGURATION USING MON_OBJ_METRICS EXTENDED
```

Setting this configuration parameter to `EXTENDED` ensures that statistics are collected for each entry in the usage list.

2. Create a usage list for the table by using the `CREATE USAGE LIST` statement. For example, to create the `INVENTORYUL` usage list for the `SALES.INVENTORY` table, issue the following command:

```
CREATE USAGE LIST INVENTORYUL FOR TABLE SALES.INVENTORY
```

3. Activate the collection of object usage statistics by using the `SET USAGE LIST STATE` statement. For example, to activate collection for the `INVENTORYUL` usage list, issue the following command:

```
SET USAGE LIST INVENTORYUL STATE = ACTIVE
```

4. During the collection of object statistics, ensure that the usage list is active and that sufficient memory is allocated for the usage list by using the `MON_GET_USAGE_LIST_STATUS` table function. For example, to check the status of the `INVENTORYUL` usage list, issue the following command:

```
SELECT MEMBER,  
       STATE,  
       LIST_SIZE,  
       USED_ENTRIES,  
       WRAPPED  
FROM TABLE(MON_GET_USAGE_LIST_STATUS('SALES', 'INVENTORYUL', -2))
```

5. When the time period for which you want to collect object usage statistics is elapsed, deactivate the collection of usage list data by using the `SET USAGE LIST STATE` statement. For example, to deactivate collection for the `INVENTORYUL` usage list, issue the following command:

```
SET USAGE LIST SALES.INVENTORYUL STATE = INACTIVE
```

6. View the information that you collected by using the `MON_GET_TABLE_USAGE_LIST` function. You can view statistics for a subset or for all of the statements that affected the table during the time period for which you collected statistics. For example, to see only the 10 statements that read the most rows of the table, issue the following command:

```
SELECT MEMBER,  
       EXECUTABLE_ID,  
       NUM_REFERENCES,  
       NUM_REF_WITH_METRICS,  
       ROWS_READ,  
       ROWS_INSERTED,  
       ROWS_UPDATED,
```

```

        ROWS_DELETED
FROM TABLE(MON_GET_TABLE_USAGE_LIST('SALES', 'INVENTORYUL', -2))
ORDER BY ROWS_READ DESC
FETCH FIRST 10 ROWS ONLY

```

7. If you want to view the text of a statement that affected the table, use the value of the **executable\_id** element in the MON\_GET\_TABLE\_USAGE\_LIST output as input for the MON\_GET\_PKG\_CACHE\_STMT table function. For example, issue the following command to view the text of a particular statement:

```

SELECT STMT_TEXT
FROM TABLE
(MON_GET_PKG_CACHE_STMT(NULL,
x'0100000000000000007C00000000000000000000020020081126171720728997',
NULL, -2))

```

8. Use the list of statements and the statistics that are provided for the statements to determine where additional monitoring or tuning, if any, is required. For example, a statement that has a low value for the **pool\_writes** monitor element compared to the **direct\_writes** monitor element value might have buffer pool issues that require attention.

## What to do next

When you do not require the information in the usage list, free the memory that is associated with the usage list by using the SET USAGE LIST STATE statement. For example, to free the memory that is associated with the INVENTORYUL usage list, issue the following command:

```
SET USAGE LIST SALES.INVENTORYUL STATE = RELEASED
```

## Identifying how a statement affects database objects

Use a section explain that includes section actuals information to identify how a statement affects database objects. You can use statistics about how the statement section affected each table or index to determine whether additional monitoring or tuning is required.

## Before you begin

Do the following tasks:

- Identify a statement for which you want to view object usage statistics.
- Ensure that you migrated your explain tables to DB2 Version 10.1.
- Ensure that automatic statistics profile generation is not enabled.
- Ensure that you have the privileges that are required to call the EXPLAIN\_FROM\_ACTIVITY procedure.

## About this task

After you identify a statement for which you want to view object usage statistics, you can get a section explain that includes section actuals information. Section actuals information indicates how the statement affected each table or index that the statement used when it executed.

Actuals information includes runtime statistics about factors like locks and buffer pool usage for each table or index. You can compare these statistics to baseline data and use them to determine where additional monitoring or tuning might be required.

## Procedure

To determine how database objects are affected by a statement:

1. Enable the collection of section actuals at the database level by issuing the following command:

```
DB2 UPDATE DATABASE CONFIGURATION USING SECTION_ACTUALS BASE
```

2. Create a workload to collect section actuals information for activities that are submitted by the application that issues the statement. For example, to create the ACTWORKLOAD workload for activities that are submitted by the TEST application and enable collection for those activities, issue the following command:

```
CREATE WORKLOAD ACTWORKLOAD APPLNAME ('TEST')  
COLLECT ACTIVITY DATA ON ALL WITH DETAILS,SECTION INCLUDE ACTUALS BASE
```

Enabling collection of section actuals can also be accomplished in the following ways:

- The CREATE SERVICE CLASS or ALTER SERVICE CLASS statement
  - The CREATE WORK ACTION SET or ALTER WORK ACTION SET statement
  - The WLM\_SET\_CONN\_ENV procedure
  - The **section\_actuals** configuration parameter
3. Create an activity event monitor by using the CREATE EVENT MONITOR statement. For example, to create the ACTEVMON activity event monitor, issue the following command:

```
CREATE EVENT MONITOR ACTEVMON  
FOR ACTIVITIES  
WRITE TO TABLE  
CONTROL (TABLE CONTROL_ACTEVMON ),  
ACTIVITY (TABLE ACTIVITY_ACTEVMON ),  
ACTIVITYSTMT (TABLE ACTIVITYSTMT_ACTEVMON ),  
ACTIVITYVALS (TABLE ACTIVITYVALS_ACTEVMON ),  
ACTIVITYMETRICS (TABLE ACTIVITYMETRICS_ACTEVMON )
```

4. Activate the activity event monitor that you created by using the SET EVENT MONITOR STATE statement. For example, to activate the ACTEVMON activity event monitor, issue the following command:

```
SET EVENT MONITOR ACTEVMON STATE 1
```

5. Run the application that issues the statement for which you want to view object statistics.
6. Find identifier information for the statement section by using the following command to query the activity event monitor tables:

```
SELECT APPL_ID,  
       UOW_ID,  
       ACTIVITY_ID,  
       STMT_TEXT  
FROM ACTIVITYSTMT_ACTEVMON
```

7. Obtain a section explain with actuals by using the activity identifier information as input for the EXPLAIN\_FROM\_ACTIVITY procedure. For example, to obtain a section explain for a section with an application ID of \*N2.DB2INST1.0B5A12222841, a unit of work ID of 16, and an activity ID of 4, issue the following command:

```
CALL EXPLAIN_FROM_ACTIVITY( '*N2.DB2INST1.0B5A12222841', 16, 4, 'ACTEVMON',  
                           'MYSHEMA', '?', '?', '?', '?', '?' )
```

You get output that looks like the following sample output:

```

Value of output parameters
-----
Parameter Name : EXPLAIN_SCHEMA
Parameter Value : MYSCHEMA

Parameter Name : EXPLAIN_REQUESTER
Parameter Value : GSDBUSER3

Parameter Name : EXPLAIN_TIME
Parameter Value : 2010-11-23-10.51.09.631945

Parameter Name : SOURCE_NAME
Parameter Value : SQLC2J21

Parameter Name : SOURCE_SCHEMA
Parameter Value : NULLID

Parameter Name : SOURCE_VERSION
Parameter Value :

```

Return Status = 0

8. Format the explain data by using the **db2exfmt** command. Use the values of the **explain\_requester**, **explain\_time**, **source\_name**, **source\_schema**, and **source\_version** parameters in the output from the EXPLAIN\_FROM\_ACTIVITY procedure as input for the command.
9. View the explain output to determine how the section affected the database objects that it used when it executed. Statistics in the output might indicate that additional monitoring or tuning is required. For example, if a table that the section uses has a high value for the **lock\_wait** monitor element, lock management might be required.
10. If you tune the statement, repeat steps 5 on page 11 through 9 to verify that performance is improved.

### What to do next

Deactivate the activity event monitor by using the SET EVENT MONITOR STATE statement. For example, to deactivate the ACTEVMON activity event monitor, issue the following command:

```
SET EVENT MONITOR ACTEVMON STATE 0
```

---

## Monitoring locking using table functions

You can retrieve information about locks using table functions. Unlike request, activity or data object monitor elements, information about locks is always available from the database manager. You do not need to enable the collection of this information.

Use the following monitor table functions to access current information for locks in the system:

- MON\_GET\_LOCKS
- MON\_GET\_APPL\_LOCKWAIT

Both table functions return data in relational form.

---

## Monitoring system memory using table functions

You can retrieve information about system memory usage using table functions.

You can examine memory usage at the level of memory sets, which are allocations of memory from the operating system. You can also examine memory usage by specific memory pools within a given memory set. Use the following monitor functions to access current information about memory usage:

- `MON_GET_MEMORY_SET`
- `MON_GET_MEMORY_POOL`

---

## Other monitoring table functions

Besides table functions that return information about the system, activities, locks, or data objects there are also table functions that return various types of miscellaneous information. These functions include ones that return information related to the fast communications manager (FCM), and about the status of table space extent movement.

Each of the table functions that follow can be used at any time. Unlike the table functions that return request metrics (the system monitoring perspective), activity metrics (the activity monitoring perspective) or metrics related to data objects (the data object monitoring perspective), it is not necessary to first enable the collection of the monitor elements returned by these functions.

- `MON_GET_FCM`
- `MON_GET_FCM_CONNECTION_LIST`
- `MON_GET_EXTENT_MOVEMENT_STATUS`

---

## Interfaces that return monitor data in XML documents

Starting in DB2 Version 9.7, some monitor data is reported as elements in XML documents.

Using XML to report monitor information provides improved extensibility and flexibility. New monitor elements can be added to the product without having to add new columns to an output table. Also, XML documents can be processed in a number of ways, depending on your needs. For example:

- You can use XQuery to run queries against the XML document.
- You can use the `XSLTRANSFORM` scalar function to transform the document into other formats.
- You can view their contents as formatted text using built-in `MON_FORMAT_XML_*` formatting functions, or the `XMLTABLE` table function.

XML documents containing monitor elements are produced by several monitoring interfaces. The sections that follow describe how results are returned as XML documents.

- “Monitor table functions with names that end with `_DETAILS`”
- “XML data returned by event monitors” on page 15.

### Monitor table functions with names that end with “`_DETAILS`”

Examples of these table functions include:

- `MON_GET_PKG_CACHE_STMT_DETAILS`
- `MON_GET_WORKLOAD_DETAILS`
- `MON_GET_CONNECTION_DETAILS`
- `MON_GET_SERVICE_SUBCLASS_DETAILS`

- MON\_GET\_ACTIVITY\_DETAILS
- MON\_GET\_UNIT\_OF\_WORK\_DETAILS

These table functions return monitor elements from the system and the activity monitoring perspectives. Most of the monitor elements returned by these functions are contained in an XML document. For example, the MON\_GET\_CONNECTION\_DETAILS table function returns the following columns:

- APPLICATION\_HANDLE
- MEMBER
- DETAILS

The DETAILS column of each row contains an XML document that contains monitor element data. This XML document is composed of several document elements that correspond to monitor elements. Figure 1 illustrates the DETAILS column containing the XML documents. In addition, it shows monitor elements returned in the XML documents in the DETAILS column.

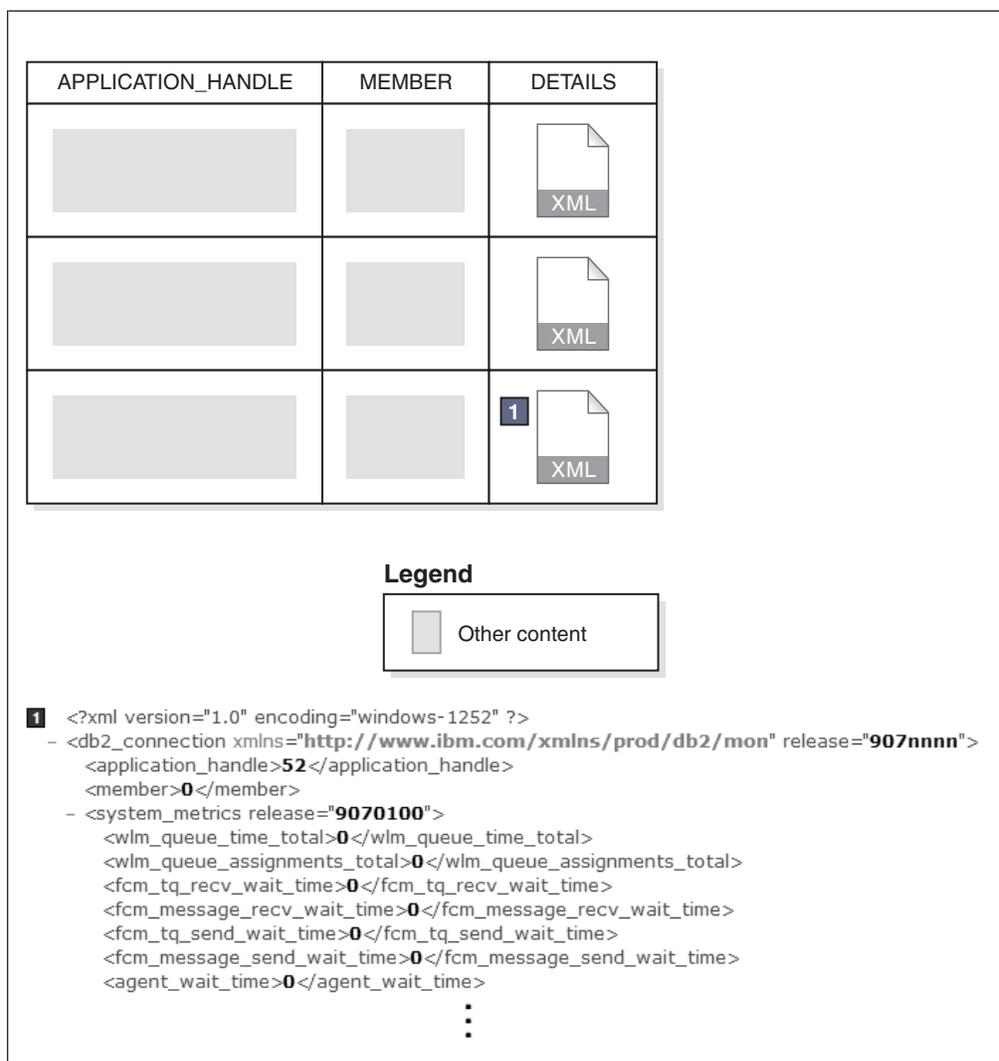


Figure 1. Table returned by MON\_GET\_CONNECTION\_DETAILS, showing the DETAILS column containing XML documents. The contents of the XML document in the third row ( **1** ) are shown following the table.

In the preceding example, the <agent\_wait\_time> XML document element corresponds to **agent\_wait\_time** monitor element.

The schema for the XML document that is returned in the DETAILS column is available in the file `sql1lib/misc/DB2MonRoutines.xsd`. Further details can be found in the file `sql1lib/misc/DB2MonCommon.xsd`.

Some of the monitor elements contained in the document in the DETAILS column might be grouped into higher-level document elements. For example, monitor elements that report on activity-related metrics are part of the **activity\_metrics** element. Similarly, system-level metrics are part of the **system\_metrics** element.

## XML data returned by event monitors

Several event monitors return data in XML format. They are summarized in Table 2. Details about the XML documents returned by the various event monitor are described in the sections that follow.

*Table 2. XML documents returned by various event monitors*

| Event monitor                            | Event monitor output format  | XML document returned   |
|--|--|---|
| "Statistics event monitor"               | <ul style="list-style-type: none"> <li>• Relational table</li> <li>• File</li> <li>• Named pipe</li> </ul> | DETAILS_XML   |
| "Activity event monitor" on page 16      | <ul style="list-style-type: none"> <li>• Relational table</li> <li>• File</li> <li>• Named pipe</li> </ul> | DETAILS_XML   |
| "Package cache event monitor" on page 17 | Unformatted event (UE) table   | METRICS<br><br>This document can be viewed only after the UE table has been transformed to either XML or relational tables. |
| "Unit of work event monitor" on page 17  | Unformatted event (UE) table   | METRICS<br><br>This document can be viewed only after the UE table has been transformed to either XML or relational tables. |

## Statistics event monitor

When you create a statistics event monitor to report on monitor elements in either the `event_scstats` and `event_wlstats` logical data groups (see "event\_scstats logical data group" on page 62, "event\_wlstats logical data group" on page 69), one of the columns produced is `DETAILS_XML`. If the event monitor is written to a table, `DETAILS_XML` is a column. If it is written to a file or named pipe, `DETAILS_XML` is part of the self-describing data stream. The document contains the **system\_metrics** monitor element, which, in turn, contains a number of monitor elements that report on metrics related to system. Figure 2 on page 16 shows the XML documents in the `DETAILS_XML` column of the table produced by the statistics event monitor:

| PARTITION_KEY | ACT_CPU_TIME_TOP | ACT_ROWS_READ_TOP | CONCURRENT_WLO_ACT_TOP | ...      | DETAILS_XML | LAST_WLM_RESET | ... |
|---------------|------------------|-------------------|------------------------|----------|-------------|----------------|-----|
|               |                  |                   |                        |          |             |                |     |
|               |                  |                   |                        |          |             |                |     |
|               |                  |                   |                        | <b>1</b> |             |                |     |

**Legend**  
 Other content

```

1 <?xml version="1.0" encoding="windows-1252" ?>
- <activity_metrics release="907nnnn" xmlns="http://www.ibm.com/xmlns/prod/db2/mon">
  <wlm_queue_time_total>0</wlm_queue_time_total>
  <wlm_queue_assignments_total>0</wlm_queue_assignments_total>
  <fcm_tq_recv_wait_time>0</fcm_tq_recv_wait_time>
  <fcm_message_recv_wait_time>0</fcm_message_recv_wait_time>
  <fcm_tq_send_wait_time>0</fcm_tq_send_wait_time>
  <fcm_message_send_wait_time>0</fcm_message_send_wait_time>
  <lock_wait_time>0</lock_wait_time>
  <lock_waits>0</lock_waits>
  <direct_read_time>0</direct_read_time>
  :
  :
  :

```

Figure 2. Output of statistics event monitor (when written to a table), showing the *DETAILS\_XML* column. The contents of the XML document in the third row (**1**) are shown following the table.

See “Information written to XML for system\_metrics and activity\_metrics monitor elements” on page 279 for the schema for the XML output from a statistics event monitor.

**Note:** system\_metrics as reported in the XML document in the *DETAILS\_XML* column produced by the statistics event monitor is also a part of the XML document contained in the *DETAILS* column returned by the *MON\_GET\_SERVICE\_SUBCLASS\_DETAILS* and *MON\_GET\_WORKLOAD\_DETAILS* table functions.

## Activity event monitor

When you create an activity event monitor to report on monitor elements in the event\_activity logical data group (see “event\_activity logical data group” on page 43), one of the columns produced is *DETAILS\_XML*. If the event monitor is written to a table, *DETAILS\_XML* is a column. If it is written to a file or named pipe, *DETAILS\_XML* is part of the self-describing data stream. Either way, the document contains the **activity\_metrics** monitor element, which, in turn, contains a number of monitor elements that report on metrics related to activities. See “Information written to XML for system\_metrics and activity\_metrics monitor elements” on page 279 for the schema for the XML output from an activity event monitor.

**Note:** activity\_metrics as reported in the XML document in the *DETAILS\_XML* column produced by the activity event monitor is also a part of the XML document contained in the *DETAILS* column returned by the *MON\_GET\_ACTIVITY\_DETAILS* table function.

## Package cache event monitor

The package cache event monitor writes its output to an unformatted event (UE) table. If you convert the data in this table using the `EVMON_FORMAT_UE_TO_TABLES` table function, one of the tables produced is `PKG_CACHE_EVENT`. This table contains a `METRICS` column. In each row, this column contains an XML document with elements associated with package cache event monitor elements.

**Note:** Starting in DB2 Version 9.7 Fix Pack 1, `EVMON_FORMAT_UE_TO_TABLES` also creates a separate table for the metrics collected by this event monitor called `PKG_CACHE_METRICS`. This table contains the same information reported in the `METRICS` column of the `PKG_CACHE_EVENT` table. So, you can retrieve metrics from the columns of the `PKG_CACHE_METRICS` table, or you can use the XML document contained in the `METRICS` column of the `PKG_CACHE_EVENT` table. See “Information written to relational tables by `EVMON_FORMAT_UE_TO_TABLES` for a package cache event monitor” on page 226 for details.

The `EVMON_FORMAT_UE_TO_XML` function also produces an XML document with elements associated with package cache event monitor elements. For example, the XML document element `<num_executions>` corresponds to the **num\_executions** monitor element. See “Information written to XML for a package cache event monitor” on page 235 for the schema for the XML output from a package cache event monitor.

## Unit of work event monitor

The unit of work event monitor writes its output to an unformatted event (UE) table. If you convert the data in this table using the `EVMON_FORMAT_UE_TO_TABLES` table function, one of the tables produced is `UOW_EVENT`. This table contains a `METRICS` column, which contains an XML document with elements associated with unit of work event monitor elements.

**Note:** Starting in DB2 Version 9.7 Fix Pack 1, `EVMON_FORMAT_UE_TO_TABLES` also creates a separate table for the metrics collected by this event monitor called `UOW_METRICS`. This table contains the same information reported in the `METRICS` column of the `UOW_EVENT` table. So, you can retrieve metrics from the columns of the `UOW_METRICS` table, or you can use the XML document contained in the `METRICS` column of the `UOW_EVENT` table. See “Information written to relational tables by `EVMON_FORMAT_UE_TO_TABLES` for a unit of work event monitor” on page 177 for details.

The `EVMON_FORMAT_UE_TO_XML` function also produces an XML document with elements associated with unit of work event monitor elements. For example, the XML document element `<workload_name>` corresponds to the **workload\_name** monitor element. See “Information written to XML by `EVMON_FORMAT_UE_TO_XML` for a unit of work event monitor” on page 188 for the schema for the XML output from a unit of work event monitor.

## Interfaces for viewing XML monitor information as formatted text

You can view the data contained in the XML documents produced by monitor interfaces in several ways, depending on how you want to view or use the data.

You can use XQuery to query and manipulate the XML documents returned by monitoring interfaces. You can also use table functions to format the XML documents for easier reading.

XQuery provides a powerful and flexible interface for querying and manipulating XML data. However, there are times where you might want to view element data in a text-based format. Depending on your needs, you can view monitor elements contained in an XML document in column- or row-oriented format. The former is useful if you know which monitor elements you want to see. The latter is useful if you do not know ahead of time which monitor elements you want to examine, such as when you want to see the top five types of wait times. The sections that follow describe two ways that you can view monitor data contained in XML documents as formatted text.

- “Viewing monitor elements in column-oriented format”
- “Viewing monitor elements in row-oriented format” on page 19

## Viewing monitor elements in column-oriented format

The XMLTABLE table function takes an XML document as input and converts it into a relational table such that each of the selected XML document elements appears as a column. This approach is useful if you know which monitor elements you want to display. For example, assume that you have created a statistics event monitor called DBSTATS to collect information from the event\_scstats logical data group. (See “event\_scstats logical data group” on page 62 for more information about the monitor elements associated with this logical data group.) The monitor elements in this logical group include **details\_xml**,<sup>1</sup> which is actually an XML document that itself contains the metrics that comprise the **system\_metrics** monitor element. (See “system\_metrics” on page 280 for more information about the monitor elements associated with the system\_metrics monitor element.) To view specific **system\_metrics** monitor elements contained in details\_xml, such as rows\_returned, total\_section\_time, or total\_cpu\_time, you can use the XMLTABLE table function to format selected monitor elements from the details\_xml documents returned by the statistics event monitor. The example that follows illustrates this. (For presentation purposes, the SQL returns results only for a specific service class.)

```
SELECT partition_number,
       service_class_id,
       statistics_timestamp,
       event.rows_returned,
       event.total_section_time,
       event.total_cpu_time
FROM   SCSTATS_DBSTATS as DBSTATS,
XMLTABLE( XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon' ),
          '$metrics/system_metrics' PASSING XMLPARSE( DOCUMENT DBSTATS.details_xml ) as "metrics"
          COLUMNS
          rows_returned          BIGINT          PATH 'rows_returned',
          total_section_time     BIGINT          PATH 'total_section_time',
          total_cpu_time         BIGINT          PATH 'total_cpu_time'
          ) AS EVENT
WHERE  service_class_id = 12;
```

The following output shows the results for this query:

---

1. Note: In these topics, when details\_xml appears in lower-case letters, it refers to the XML document details\_xml. DETAILS\_XML, in upper-letters, refers to a column in a relational table called DETAILS\_XML that contains the details\_xml documents.

| PARTITION_NUMBER | SERVICE_CLASS_ID | STATISTICS_TIMESTAMP       | ROWS_RETURNED | TOTAL_SECTION_TIME | TOTAL_CPU_TIME |
|------------------|------------------|----------------------------|---------------|--------------------|----------------|
| 0                | 12               | 2010-01-05-12.14.37.001717 | 402           | 990                | 1531250        |
| 0                | 12               | 2010-01-05-12.15.00.035409 | 402           | 990                | 1531250        |
| 0                | 12               | 2010-01-05-12.20.00.021884 | 412           | 1064               | 1609375        |
| 0                | 12               | 2010-01-05-12.25.00.039175 | 422           | 1075               | 1687500        |
| 0                | 12               | 2010-01-05-12.29.59.950137 | 432           | 1104               | 1765625        |
| 0                | 12               | 2010-01-05-12.34.59.948979 | 442           | 1130               | 1796875        |
| 0                | 12               | 2010-01-05-12.39.59.903928 | 452           | 1149               | 1890625        |
| 0                | 12               | 2010-01-05-12.44.59.953596 | 462           | 1178               | 1953125        |
| 0                | 12               | 2010-01-05-12.49.59.970059 | 473           | 1207               | 2062500        |
| 0                | 12               | 2010-01-05-12.54.59.971990 | 483           | 1230               | 2109375        |

10 record(s) selected.

In this case, the first three columns are displayed directly from the table SCSTATS\_DBSTATS table produced by the statistics event monitor. The last three columns are metrics monitor elements extracted from the XML document in the DETAILS\_XML column of the table.

For more information about using XMLTABLE, refer to the documentation for that function. You can also see examples of using XMLTABLE to view monitor elements in the documentation for the various MON\_GET\_\*\_DETAILS functions.

### Viewing monitor elements in row-oriented format

The table functions with names of the form MON\_FORMAT\_XML\_\*\_BY\_ROW introduced in DB2 Version 9.7 Fix Pack 1 provide a quick way to display the metrics monitor elements contained in an XML document. They report metrics in a row-based format, with each monitor element appearing in a row by itself. The following functions are included in this group:

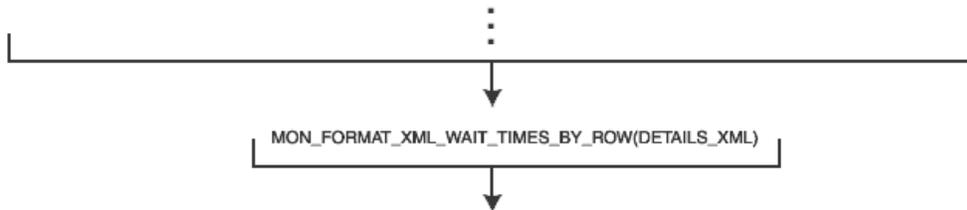
- MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW
- MON\_FORMAT\_XML\_TIMES\_BY\_ROW
- MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW
- MON\_FORMAT\_XML\_METRICS\_BY\_ROW

For example, the XML document returned by the statistics event monitor, DETAILS\_XML, might look something like the one shown in the first part of Figure 3 on page 20. If you use the MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW function to format the content of DETAILS\_XML, the output would look like the table at the bottom of the diagram.

```

<?xml version="1.0" encoding="windows-1252" ?>
- <system_metrics xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="907nnnn">
  <wlm_queue_time_total>0</wlm_queue_time_total>
  <wlm_queue_assignments_total>0</wlm_queue_assignments_total>
  <fcm_tq_rcv_wait_time>0</fcm_tq_rcv_wait_time>
  <fcm_message_rcv_wait_time>0</fcm_message_rcv_wait_time>
  <fcm_tq_send_wait_time>0</fcm_tq_send_wait_time>
  <fcm_message_send_wait_time>0</fcm_message_send_wait_time>
  <agent_wait_time>0</agent_wait_time>
  <agent_waits_total>0</agent_waits_total>
  <lock_wait_time>0</lock_wait_time>

```



| METRIC_NAME                | TOTAL_TIME_VALUE | COUNT | PARENT_METRIC_NAME |
|----------------------------|------------------|-------|--------------------|
| WLM_QUEUE_TIME_TOTAL       | 0                | 0     | TOTAL_WAIT_TIME    |
| FCM_TQ_RECV_WAIT_TIME      | 0                | 0     | FCM_RECV_WAIT_TIME |
| FCM_MESSAGE_RECV_WAIT_TIME | 0                | 0     | FCM_RECV_WAIT_TIME |
| FCM_TQ_SEND_WAIT_TIME      | 0                | 0     | FCM_SEND_WAIT_TIME |
| FCM_MESSAGE_SEND_WAIT_TIME | 0                | 0     | FCM_SEND_WAIT_TIME |
| AGENT_WAIT_TIME            | 0                | 0     | TOTAL_WAIT_TIME    |
| LOCK_WAIT_TIME             | 0                | 0     | TOTAL_WAIT_TIME    |
| DIRECT_READ_TIME           | 0                | 0     | TOTAL_WAIT_TIME    |
| DIRECT_WRITE_TIME          | 0                | 0     | TOTAL_WAIT_TIME    |
| LOG_BUFFER_WAIT_TIME       | 0                | 0     | TOTAL_WAIT_TIME    |
| LOG_DISK_WAIT_TIME         | 0                | 0     | TOTAL_WAIT_TIME    |
| ⋮                          |                  |       |                    |

Figure 3. An XML file containing monitoring data, processed by one of the MON\_FORMAT\_XML\_\* functions. This example shows the use of the MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW function. Only wait times are returned; other metrics contained in the XML file, such as wlm\_queue\_assignments\_total are excluded by this particular function.

The number of columns returned varies by the specific function that you use. For example MON\_FORMAT\_XML\_METRICS\_BY\_ROW returns two columns, one for the metric name, and one for its corresponding value:

| METRIC_NAME               | VALUE |
|---------------------------|-------|
| WLM_QUEUE_TIME_TOTAL      | 0     |
| WLM_QUEUE_ASSIGNMENTS_TOT | 0     |
| FCM_TQ_RECV_WAIT_TIME     | 0     |
| FCM_MESSAGE_RECV_WAIT_TIM | 0     |
| FCM_TQ_SEND_WAIT_TIME     | 0     |
| ⋮                         |       |

By comparison, MON\_FORMAT\_XML\_TIMES\_BY\_ROW returns four columns:

| METRIC_NAME                | TOTAL_TIME_VALUE | COUNT | PARENT_METRIC_NAME |
|----------------------------|------------------|-------|--------------------|
| WLM_QUEUE_TIME_TOTAL       | 0                | 0     | TOTAL_WAIT_TIME    |
| FCM_TQ_RECV_WAIT_TIME      | 0                | 0     | FCM_RECV_WAIT_TIME |
| FCM_MESSAGE_RECV_WAIT_TIME | 0                | 0     | FCM_RECV_WAIT_TIME |
| FCM_TQ_SEND_WAIT_TIME      | 0                | 0     | FCM_SEND_WAIT_TIME |
| FCM_MESSAGE_SEND_WAIT_TIME | 0                | 0     | FCM_SEND_WAIT_TIME |
| ⋮                          |                  |       |                    |

The MON\_FORMAT\_XML\_\*\_BY\_ROW functions are useful when you do not know which elements you want to view. For example, you might want to see the

top 10 wait-time monitor elements for the workload named CLPWORKLOAD. To collect this information, you can create a statistics event monitor called DBSTATS (event\_wlstats logical data group). Assuming you set up this event monitor to write to a table, it records metrics in a column called DETAILS\_XML. Once the output table from the event monitor is populated with monitor data, you can construct a query that uses the MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW function to extract the monitor elements you want to see:

```
SELECT SUBSTR(STATS.WORKLOAD_NAME,1,15) AS WORKLOAD_NAME,
       SUBSTR(METRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(METRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE
FROM   WLSTATS_DBSTATS AS STATS,
       TABLE(MON_FORMAT_XML_WAIT_TIMES_BY_ROW(STATS.DETAILS_XML)) AS METRICS
WHERE  WORKLOAD_NAME='CLPWORKLOAD' AND (PARENT_METRIC_NAME='TOTAL_WAIT_TIME')
GROUP BY WORKLOAD_NAME,METRIC_NAME
ORDER BY TOTAL_TIME_VALUE DESC
FETCH FIRST 10 ROWS ONLY
```

**Remember:** Time spent monitor elements are organized into hierarchies. In this example, to avoid double-counting wait times, only the monitor elements that roll-up to **total\_wait\_time** are included (see the WHERE clause in the preceding SQL statement). Otherwise, **total\_wait\_time** itself would be included in the results, which includes several individual wait times.

The output that follows shows what the results of the preceding query might look like:

| WORKLOAD_NAME | METRIC_NAME             | TOTAL_TIME_VALUE |
|---------------|-------------------------|------------------|
| CLPWORKLOAD   | LOCK_WAIT_TIME          | 15138541         |
| CLPWORKLOAD   | DIRECT_READ_TIME        | 6116231          |
| CLPWORKLOAD   | POOL_READ_TIME          | 6079458          |
| CLPWORKLOAD   | DIRECT_WRITE_TIME       | 452627           |
| CLPWORKLOAD   | POOL_WRITE_TIME         | 386208           |
| CLPWORKLOAD   | IPC_SEND_WAIT_TIME      | 283172           |
| CLPWORKLOAD   | LOG_DISK_WAIT_TIME      | 103888           |
| CLPWORKLOAD   | DIAGLOG_WRITE_WAIT_TIME | 78198            |
| CLPWORKLOAD   | IPC_RECV_WAIT_TIME      | 15612            |
| CLPWORKLOAD   | TCPIP_SEND_WAIT_TIME    | 3291             |

10 record(s) selected.

**Note:** The MON\_FORMAT\_XML\_\*\_BY\_ROW functions return only monitor elements that track measurements or *metrics*. These include monitor elements that track wait and component times, as well as counters. They do not return non-metrics monitor elements contained in the XML document, such as uow\_id, or activity\_id.

You can use the XMLTABLE function to view *any* of the elements (including non-metrics elements) contained in the XML document. However, the most frequently used, non-metrics monitor elements are returned as columns by the monitor functions that begin with MON\_GET\_\*, such as MON\_GET\_UNIT\_OF\_WORK, or MON\_GET\_CONNECTION. If you are not familiar with XML, you might find it faster and easier to create queries using these functions than using the XMLTABLE function to extract monitor elements from an XML document.

To summarize: if you are interested in viewing non-metrics monitor elements, the MON\_GET\_\* series of table functions might be a good alternative to the XMLTABLE function. If you are interested in viewing metrics monitor elements, the MON\_FORMAT\_XML\_\*\_BY\_ROW table functions might suit your needs.

## Viewing metrics monitor elements from XML documents as rows in a table

One way to view metrics-related information contained in an XML document returned from an event monitor is to convert it into a format where each monitor element appears in a row by itself. This format is useful if you want to view the information in a text-based format, but do not know specifically which monitor elements you want to examine.

### About this task

To view metrics information in row-based format from the XML documents returned by various monitoring interfaces, use the `MON_FORMAT_XML_*_BY_ROW` table functions. These functions were introduced in DB2 Version 9.7 Fix Pack 1.

### Procedure

The example shown in this task uses the `MON_FORMAT_XML_TIMES_BY_ROW` table function to view component times for a statement as tracked by the package cache event monitor. It assumes that a package cache event monitor called `PKG_CACHEEVENTS` has been created and activated. The package cache event monitor writes its output to an unformatted event (UE) table. Before it can be used, the data in the UE table must be converted to either relational tables using the `EVMON_FORMAT_UE_TO_TABLES` stored procedure, or to XML using the `EVMON_FORMAT_UE_TO_XML` table function. This task shows the first of these two approaches.

1. First, convert the unformatted event (UE) table that the package cache event monitor writes to into relational tables using the `EVMON_FORMAT_UE_TO_TABLES` procedure

```
ca11 EVMON_FORMAT_UE_TO_TABLES ('PkgCache',NULL,NULL,NULL,NULL,NULL,
    NULL,0,'SELECT * FROM PKGCACHEEVENTS')
```

This procedure creates two tables:

- One is called `PKG_CACHE_EVENT`, which contains a column called `METRICS`. This column, in turn, contains XML documents with metrics monitor elements.
- The other is called `PKG_CACHE_METRICS`.

**Note:** You could view the metrics directly from the columns in `PKG_CACHE_METRICS`, rather than extract metrics from the `METRICS` column of the `PKG_CACHE_EVENT` table. However, when you examine `PKG_CACHE_METRICS`, the metrics appear in columns, rather than rows; it is not as easy to get a ranking of, say, the metrics with the highest values.

2. Query the two tables produced in the preceding step to determine which statement is the most expensive in terms of execution times:

```
SELECT EVENTS.EXECUTABLE_ID,
       SUM(METRICS.STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME
FROM   PKGCACHE_EVENT AS EVENTS,
       PKGCACHE_METRICS AS METRICS
WHERE  EVENTS.XMLID = METRICS.XMLID
GROUP BY EVENTS.EXECUTABLE_ID
ORDER BY TOTAL_STMT_EXEC_TIME DESC
FETCH FIRST 5 ROWS ONLY
```

In the preceding query, the two tables produced in step 1 on page 22 are joined so that the statement IDs from the PKGCACHE\_EVENT table can be associated with their execution times in the PKGCACHE\_METRICS table:

| EXECUTABLE_ID  | TOTAL_STMT_EXEC_TIME |
|--|----------------------|
| x'0100000000000001A03000000000000000000000020020091215115933859000'  | 250                  |
| x'01000000000000015030000000000000000000000020020091215115850328000' | 191                  |
| x'01000000000000021020000000000000000000000020020091215115818343001' | 129                  |
| x'010000000000000C4020000000000000000000000020020091215115838578000' | 41                   |
| x'010000000000000B0020000000000000000000000020020091215115838203000' | 38                   |

5 record(s) selected.

The first item in the results represents the statement with the largest overall execution time.

- Optional: If you like, you can display the text for the statement using the following SQL:

```
SELECT SUBSTR(STMT_TEXT,1,60) AS STMT_TEXT
FROM PKGCACHE_EVENT
WHERE EXECUTABLE_ID = x'0100000000000001A03000000000000000000000020020091215115933859000'
```

Results:

| STMT_TEXT  |
|--|
| DROP XSROBJECT MYSCHEMA.EVMON_PKGCACHE_SCHEMA_SQL09070 |

1 record(s) selected.

- Use the MON\_FORMAT\_XML\_TIMES\_BY\_ROW table function to view a listing of the time-spent monitor elements for the statement you identified in step 2 on page 22:

```
SELECT SUBSTR(XMLMETRICS.METRIC_NAME,1,30) AS METRIC_NAME,
XMLMETRICS.TOTAL_TIME_VALUE,
SUBSTR(XMLMETRICS.PARENT_METRIC_NAME,1,30) AS PARENT_METRIC_NAME
FROM PKGCACHE_EVENT AS EVENTS,
TABLE(MON_FORMAT_XML_TIMES_BY_ROW(EVENTS.METRICS)) AS XMLMETRICS
WHERE EVENTS.EXECUTABLE_ID=
x'0100000000000001A03000000000000000000000020020091215115933859000'
AND PARENT_METRIC_NAME='STMT_EXEC_TIME'
ORDER BY XMLMETRICS.TOTAL_TIME_VALUE DESC
```

Notes:

- Remember that time-spent monitor elements are organized into hierarchies. To eliminate double-counting, only those metrics that roll-up to **stmt\_exec\_time** are included in the results. Otherwise, **stmt\_exec\_time** itself would be included in the results, which includes several individual component times.
- PARENT\_METRIC\_NAME, one of the columns returned by MON\_FORMAT\_XML\_TIMES\_BY\_ROW is included for illustrative purposes.

When run, the following results are returned by this query:

| METRIC_NAME             | TOTAL_TIME_VALUE | PARENT_METRIC_NAME |
|-------------------------|------------------|--------------------|
| TOTAL_ACT_WAIT_TIME     | 234              | STMT_EXEC_TIME     |
| TOTAL_SECTION_PROC_TIME | 15               | STMT_EXEC_TIME     |

Here, you can see that the total processing time adds up to 249 ms. Compare this time to the total time of 250 shown in step 2 on page 22; the extra millisecond is accounted for by other times (for example, waits) not included in **stmt\_exec\_time**.

## Results

In the results from the preceding example, you can see the arrangement of the metrics: they appear in row-oriented format, one metric per row. The advantage of using this approach is that you do not need to know ahead of time which metrics or monitor elements you want to see. If you want to see which of the time-spent metrics have the five highest values, or which metrics fall within a specific range of values, you can easily create a query to return the results you are interested in. By contrast, if you use the XMLTABLE function to display the monitor elements as columns, you need to specify which monitor elements to display (or display them all).

## Example

*Viewing the contents of the DETAILS column produced by a MON\_GET\_\*\_DETAILS table function*

You can also use the MON\_FORMAT\_XML\_\*\_BY\_ROW functions to view the contents of the DETAILS column returned by any of the MON\_GET\_\*\_DETAILS functions. For example, MON\_GET\_CONNECTION\_DETAILS returns a DETAILS column that contains an XML document with metrics that pertain to a database connection.

For example, to view the non-zero component times for each connection across all members, you could use the following query:

```
SELECT CONDETAILS.APPLICATION_HANDLE,
       SUBSTR(XMLMETRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(XMLMETRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE,
       SUBSTR(XMLMETRICS.PARENT_METRIC_NAME,1,30) AS PARENT_METRIC_NAME
FROM TABLE(MON_GET_CONNECTION_DETAILS(NULL,-1)) AS CONDETAILS,
     TABLE(MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW(CONDETAILS.DETAILS))AS XMLMETRICS
WHERE TOTAL_TIME_VALUE > 0 AND XMLMETRICS.PARENT_METRIC_NAME='TOTAL_RQST_TIME'
GROUP BY CONDETAILS.APPLICATION_HANDLE,
         XMLMETRICS.PARENT_METRIC_NAME,
         XMLMETRICS.METRIC_NAME
ORDER BY CONDETAILS.APPLICATION_HANDLE ASC, TOTAL_TIME_VALUE DESC
```

### Notes:

- To eliminate double-counting, only those metrics that roll-up to **total\_rqst\_time** are included in the results (WHERE .... XMLMETRICS.PARENT\_METRIC\_NAME='TOTAL\_RQST\_TIME'). Otherwise, total\_rqst\_time itself would be included in the results, which includes several individual component times.
- PARENT\_METRIC\_NAME, one of the columns returned by MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW is included for illustrative purposes.

The preceding query returns the following results:

| APPLICATION_HANDLE | METRIC_NAME                 | TOTAL_TIME_VALUE | PARENT_METRIC_NAME |
|--------------------|-----------------------------|------------------|--------------------|
| 52                 | TOTAL_SECTION_TIME          | 3936             | TOTAL_RQST_TIME    |
| 52                 | TOTAL_COMPILE_TIME          | 482              | TOTAL_RQST_TIME    |
| 52                 | TOTAL_COMMIT_TIME           | 15               | TOTAL_RQST_TIME    |
| 52                 | TOTAL_ROLLBACK_TIME         | 1                | TOTAL_RQST_TIME    |
| 496                | TOTAL_COMPILE_TIME          | 251              | TOTAL_RQST_TIME    |
| 496                | TOTAL_SECTION_TIME          | 46               | TOTAL_RQST_TIME    |
| 496                | TOTAL_IMPLICIT_COMPILE_TIME | 5                | TOTAL_RQST_TIME    |

7 record(s) selected.

As this example shows, only metrics that comprise **total\_rqst\_time** are included. Had the WHERE .... XMLMETRICS.PARENT\_METRIC\_NAME='TOTAL\_RQST\_TIME' clause not been included in the query, the results would look like those that follow:

| APPLICATION_HANDLE | METRIC_NAME                    | TOTAL_TIME_VALUE | PARENT_METRIC_NAME        |
|--------------------|--------------------------------|------------------|---------------------------|
|                    | <b>52 TOTAL_RQST_TIME</b>      | <b>4603</b>      | -                         |
| 52                 | TOTAL_SECTION_TIME             | 3942             | TOTAL_RQST_TIME           |
| 52                 | TOTAL_COMPILE_TIME             | 537              | TOTAL_RQST_TIME           |
| 52                 | <i>TOTAL_SECTION_SORT_TIME</i> | 299              | <i>TOTAL_SECTION_TIME</i> |
| 52                 | TOTAL_COMMIT_TIME              | 15               | TOTAL_RQST_TIME           |
| 52                 | TOTAL_ROLLBACK_TIME            | 1                | TOTAL_RQST_TIME           |
|                    | <b>496 TOTAL_RQST_TIME</b>     | <b>341</b>       | -                         |
| 496                | TOTAL_COMPILE_TIME             | 251              | TOTAL_RQST_TIME           |
| 496                | TOTAL_SECTION_TIME             | 46               | TOTAL_RQST_TIME           |
| 496                | TOTAL_IMPLICIT_COMPILE_TIME    | 5                | TOTAL_RQST_TIME           |
| 496                | <i>TOTAL_SECTION_SORT_TIME</i> | 2                | <i>TOTAL_SECTION_TIME</i> |

11 record(s) selected.

In this case, the values for **total\_rqst\_time** for each connection are included in the results, which includes the values for all other elements for which it is the parent. Similarly, the values for items in *italics* roll up to the **total\_section\_time**. Had they not been excluded in the WHERE clause, they would have been triple-counted in the results, as **total\_section\_time** itself rolls up to **total\_rqst\_time**.



---

## Chapter 3. Event monitors

Monitoring table functions and snapshot routines return the values of monitor elements at the specific point in time the routine is run, which is useful when you want to check the current state of your system. However, there are many times when you need to capture information about the state of your system at exactly the time that a specific event occurs. Event monitors serve this purpose.

Event monitors can be created to capture point-in-time information related to different kinds of *events* that take place in your system. For example, you can create an event monitor to capture information when a specific threshold that you define is exceeded. The information captured includes such things as the ID of the application that was running when the threshold was exceeded. Or, you might create an event monitor to determine what statement was running when a lock event occurred.

---

### Types of events for which event monitors capture data

You can use event monitors to capture information related to many different kinds of events that take place on your system.

The following tables lists the types of events that occur in the system that you can monitor with an event monitor. It also describes the type of data collected for different events, as well as when the monitoring data is collected. The names of the event monitors shown in column two correspond to the keywords used to create that type of event monitor using the CREATE EVENT MONITOR statement.

Table 3. Event Types

| Type of event to monitor | Event monitor name | Event monitor properties                            | Details   |
|--------------------------|--------------------|---|---|
| Locks and deadlocks      | LOCKING            | <i>Uses of this event monitor</i>                   | To determine when locks or deadlocks occur, and the applications that are involved. The advantages of using the LOCKING event monitor instead of the deprecated DEADLOCKS event monitor include consolidated reporting of both lock and deadlock events, as well as the inclusion of information about lock waits and lock time-outs. |
|                          |                    | <i>Data collected</i>                               | Comprehensive information regarding applications involved, including the identification of participating statements (and statement text) and a list of locks being held.  |
|                          |                    | <i>When the event data is generated<sup>1</sup></i> | Upon detection of any of the following event types, depending on how you configure the event monitor: <ul style="list-style-type: none"><li>• lock timeout</li><li>• deadlock</li><li>• lock wait beyond a specified duration</li></ul>   |

Table 3. Event Types (continued)

| Type of event to monitor  | Event monitor name | Event monitor properties                        | Details   |
|---|--------------------|---|---|
| Execution of a SQL statements or other operation that spawns a database activity. | ACTIVITIES         | <i>Uses of this event monitor</i>               | To track the execution of individual statements and other activities to understand what activities are running in the system. Also to capture activities for diagnostic reasons, and to study the resource consumption of SQL.  |
|   |                    | <i>Data collected</i>                           | <p>Activity level data, generally for activities involving workload management objects.</p> <ul style="list-style-type: none"> <li>If WITH DETAILS was specified as part of COLLECT ACTIVITY DATA clause on the CREATE or ALTER statements for a workload management object, then information collected includes statement and compilation environment information for those activities that have it. If WITH SECTION is also specified, then statement, compilation environment, section environment data, and section actuals are also captured.</li> <li>If AND VALUES was also specified on the CREATE OR ALTER statement for the workload management object, the information collected will also include input data values for those activities that have it.</li> </ul> |
|   |                    | <i>When event data is generated<sup>1</sup></i> | <ul style="list-style-type: none"> <li>Upon completion of an activity that executed in a service class, workload or work class that had its COLLECT ACTIVITY DATA option turned on.</li> <li>When an activity violates a threshold that has the COLLECT ACTIVITY DATA option enabled.</li> <li>At the instant the WLM_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure is executed.</li> <li>When an activity is executed by a connection for which activity collection has been enabled using the WLM_SET_CONN_ENV stored procedure.</li> </ul>   |
| Execution of an SQL statement   | STATEMENTS         | <i>Uses of this event monitor</i>               | To see what requests are being made to the database as a result of the execution of SQL statements.   |
|   |                    | <i>Data collected</i>                           | <p>Statement start or stop time, CPU used, text of dynamic SQL, SQLCA (return code of SQL statement), and other metrics such as fetch count. For partitioned databases: CPU used, execution time, table and table queue information.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>When monitoring the execution of SQL procedures using statement event monitors, data manipulation language (DML) statements, such as INSERT, SELECT, DELETE, and UPDATE, generate events. Procedural statements, such as variable assignments and control structures (for example, WHILE or IF), do not generate events in a deterministic fashion.</li> <li>Statement start or stop time is unavailable when the Timestamp switch is off.</li> </ul>                    |
|   |                    | <i>When event data is generated</i>             | End of SQL statement <sup>2</sup> ; for partitioned databases, End of subsection <sup>2</sup>   |

Table 3. Event Types (continued)

| Type of event to monitor                    | Event monitor name         | Event monitor properties                         | Details   |
|---|----------------------------|--|---|
| Completion of a unit of work (transaction)  | UNIT OF WORK               | <i>Uses of this event monitor</i>                | To gather resource usage information and performance metrics for units of work that run on the system. This information can be used for purposes ranging from generating reports for billing or charge-back purposes of system resources used by an application, to troubleshooting performance problems caused by slow-running routines.<br><br>Recommended over the TRANSACTIONS event monitor. |
|   |                            | <i>Data collected</i>                            | Information about units of work (transactions), such as start and stop time, the workload and service class under which they ran. Option to include information about packages or executable IDs for statements run as part of the unit of work, as well as request metrics.  |
|   |                            | <i>When event data is generated</i> <sup>1</sup> | Upon completion of a unit of work   |
| Eviction of sections from the package cache | PACKAGE CACHE              | <i>Uses of this event monitor</i>                | To capture a history of statements (and related metrics) that are no longer in the package cache. This information can be used if you need to examine performance metrics for statements that are no longer available in memory.  |
|   |                            | <i>Data collected</i>                            | Includes statement text and metrics aggregated over all executions of the section.  |
|   |                            | <i>When event data is generated</i> <sup>1</sup> | As entries are evicted from the package cache.  |
| Connections to the database by applications | CONNECTIONS                | <i>Uses of this event monitor</i>                | To capture metrics and other monitor elements for each connection to the database by an application.  |
|   |                            | <i>Data collected</i>                            | All application-level counters. For example, the time that the application connected to or disconnected from the database, or number of lock escalations that the application was involved with.  |
|   |                            | <i>When event data is generated</i>              | End of connection <sup>2</sup>  |
| Deactivation of database                    | DATABASE                   | <i>Uses of this event monitor</i>                | To capture metrics and other monitor elements that reflect information about the database as whole, since activation.   |
|   |                            | <i>Data collected</i>                            | All database level counters. For example, the number of connections made to a database, time spent waiting on locks, or rows of data inserted since its activation.   |
|   |                            | <i>When event data is generated</i>              | Database deactivation <sup>2</sup>  |
|   | BUFFERPOOLS<br>TABLESPACES | <i>Uses of this event monitor</i>                | To capture metrics related to buffer pools and table spaces.  |
|   |                            | <i>Data collected</i>                            | Counters for buffer pools, prefetchers, page cleaners and direct I/O for each buffer pool.  |
|   |                            | <i>When event data is generated</i>              | Database deactivation <sup>2</sup>  |
|   | TABLES                     | <i>Uses of this event monitor</i>                | To capture metrics related to tables that have changed since database activation.   |
|   |                            | <i>Data collected</i>                            | Table level counters, such as rows read or written, or disk pages used by data,LOB or index objects.  |
|   |                            | <i>When event data is generated</i>              | Database deactivation <sup>2</sup>  |

Table 3. Event Types (continued)

| Type of event to monitor   | Event monitor name   | Event monitor properties                        | Details   |
|--|----------------------|---|---|
| Statistics and metrics on workload management objects  | STATISTICS           | <i>Uses of this event monitor</i>               | To capture processing metrics related to workload management objects (for example service superclasses, or workloads) in the database. For example, you could use a statistics event monitor to check on CPU utilization over time for a given workload.  |
|  |                      | <i>Data collected</i>                           | Statistics computed from the activities that executed within each service class, workload, or work class that exists on the system.   |
|  |                      | <i>When event data is generated</i>             | Statistics can be collected automatically at regular intervals. This interval is defined with the <b>wlm_collect_int</b> database configuration parameter.<br><br>Data can also be collected manually, using the WLM_COLLECT_STATS stored procedure.<br><b>Note:</b> With either collection mechanism, the values of statistics monitor elements are reset to 0 after collection has taken place.   |
| Exceeding a workload manager threshold   | THRESHOLD VIOLATIONS | <i>Uses of this event monitor</i>               | To determine when specific thresholds that you set are exceeded during database operations. Thresholds can be set for a variety of things, ranging from CPU time to the number of database connections, to the execution of specific statements. Data collected can be used for a variety of purposes, including monitoring for potential problems (such as approaching limits on temporary table space).   |
|  |                      | <i>Data collected</i>                           | Threshold violation information.  |
|  |                      | <i>When event data is generated</i>             | Upon detection of a threshold violation. Thresholds are defined using the CREATE THRESHOLD statement.   |
| Changes to database or database manager configuration  | CHANGE HISTORY       | <i>Uses of this event monitor</i>               | To capture change to database and database manager configuration and registry settings, execution of DDL statements, and execution of utilities   |
|  |                      | <i>Data collected</i>                           | Database and database manager configuration parameter changes, registry variable changes, execution of DDL statements, execution of certain DB2 utilities and commands, and change history event monitor startup.<br><b>Note:</b> Generally, information related to events that occur while the change history event monitor is inactive or the database is offline are not captured. However, changes to registry variables and configuration parameters are recorded. |
|  |                      | <i>When event data is generated<sup>1</sup></i> | Upon monitor startup, when a parameter or variable changes, or when a command, DDL, or utility completes.   |
| <b>Notes:</b>  |                      |   |   |
| <ol style="list-style-type: none"> <li>1. If a database is deactivated while an activity event monitor is active, backlogged activity records in the queue are discarded. To ensure that you obtain all activities event monitor records and that none are discarded, deactivate the activities event monitor before deactivating the database. When an activities event monitor is explicitly deactivated, all backlogged activity records in the queue are processed before the event monitor deactivates.</li> <li>2. In addition to the defined times where data collection automatically occurs, you can use the FLUSH EVENT MONITOR SQL statement to generate events. The events generated by this method are written with the current database monitor values for all the monitor types (except for DEADLOCKS and DEADLOCKS WITH DETAILS) associated with the flushed event monitor.</li> </ol> |                      |   |   |

Table 4. Event Types For Deprecated Event Monitors

| Type of event to monitor  | Event monitor name                          | Event monitor properties  | Details   |
|---|---|---|---|
| Deadlocks   | DEADLOCKS <sup>2</sup>                      | <i>Uses of this event monitor</i>   | To determine when deadlocks occur, and the applications that are involved.  |
|   |   | <i>Data collected</i>   | Applications involved, and locks in contention.   |
|   |   | <i>When event data is generated</i>   | Detection of a deadlock   |
|   | DEADLOCKS WITH DETAILS <sup>2</sup>         | <i>Uses of this event monitor</i>   | To determine when deadlocks occur, and the applications that are involved.  |
|   |   | <i>Data collected</i>   | Comprehensive information regarding applications involved, including the identification of participating statements (and statement text) and a list of locks being held. Using a DEADLOCKS WITH DETAILS event monitor instead of a DEADLOCKS event monitor will incur a performance cost when deadlocks occur, due to the extra information that is collected.  |
|   |   | <i>When event data is generated</i>   | Detection of a deadlock   |
|   | DEADLOCKS WITH DETAILS HISTORY <sup>2</sup> | <i>Uses of this event monitor</i>   | To determine when deadlocks occur, and the applications that are involved.  |
|   |   | <i>Data collected</i>   | All information reported in a DEADLOCKS WITH DETAILS event monitor, along with the statement history for the current unit of work of each application owning a lock participating in a deadlock scenario for the database partition where that lock is held. Using a DEADLOCKS WITH DETAILS HISTORY event monitor will incur a minor performance cost when activated due to statement history tracking. |
|   |   | <i>When event data is generated</i>   | Detection of a deadlock   |
| DEADLOCKS WITH DETAILS HISTORY VALUES <sup>2</sup>  | <i>Uses of this event monitor</i>           |   |   |
|   | <i>Data collected</i>                       | All information reported in a deadlock with details and history, along with the values provided for any parameter markers at the time of execution of a statement. Using a DEADLOCKS WITH DETAILS HISTORY VALUES event monitor will incur a more significant performance cost when activated due to extra copying of data values. |   |
|   | <i>When event data is generated</i>         | Detection of a deadlock   |   |
| Completion of a unit of work (transaction)  | TRANSACTIONS <sup>3</sup>                   | <i>Uses of this event monitor</i>   |   |
|   |   | <i>Data collected</i>   | UOW work start or stop time, previous UOW time, CPU consumed, locking and logging metrics. Transaction records are not generated if running with XA.  |
|   |   | <i>When event data is generated</i>   | Upon completion of a unit of work <sup>1</sup>  |
| <b>Notes:</b>   |   |   |   |
| 1. In addition to the defined times where data collection automatically occurs, you can use the FLUSH EVENT MONITOR SQL statement to generate events. The events generated by this method are written with the current database monitor values for all the monitor types (except for DEADLOCKS and DEADLOCKS WITH DETAILS) associated with the flushed event monitor. |   |   |   |
| 2. This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.  |   |   |   |
| 3. This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR UNIT OF WORK statement to monitor transaction events.  |   |   |   |

**Note:** A detailed deadlock event monitor is created for each newly created database. This event monitor, named DB2DETAILDEADLOCK, starts when the database is activated and will write to files in the database directory. You can avoid the additional processor time this event monitor requires by dropping it. The DB2DETAILDEADLOCK event monitor is deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

---

## Working with event monitors

Generally, the process of creating and using event monitors to capture information about the system when certain events occur is similar for all event monitor types. First you create the event monitor, then you enable data collection, and finally, you access the data gathered.

### About this task

This topic provides an outline of the general steps to follow when working with event monitors.

### Procedure

To use an event monitor to capture event information:

1. Create the event monitor. To create an event monitor, use the appropriate version of the CREATE EVENT MONITOR statement. When you create an event monitor, you must choose how to record the data the event monitor collects. All event monitors can write their output to relational tables; however, depending on your specific purposes, there are different options that might be more appropriate.
2. Activate the event monitor. To activate the event monitor, use the SET EVENT MONITOR STATE statement. For example, for an event monitor called **capturestats**, use the following command:

```
SET EVENT MONITOR capturestats STATE 1
```

To turn off data collection by the event monitor, use the following statement:

```
SET EVENT MONITOR capturestats STATE 0
```

By default, some event monitors activate automatically upon database activation; others require that you activate them manually. However, an event monitor created with the AUTOSTART option will not automatically be activated until the next database activation. Use the SET EVENT MONITOR STATE statement to force a recently-created event monitor into the active state. To determine whether an event monitor starts automatically, refer to the reference information for the relevant CREATE EVENT MONITOR statement.

3. Enable the collection of data. (Only for LOCKING, ACTIVITIES, STATISTICS, UNIT OF WORK and PACKAGE CACHE event monitors) Enabling data collection involves configuring the database manager to gather specific types of data to be recorded by event monitors.

Not all event monitors require data collection to be enabled; for those that do not, such as the TABLE event monitor, creating and activating them is sufficient to cause data to be collected. The threshold violations event monitor also starts data collection automatically; however, in this case, you must also define the thresholds for which you want data captured using the CREATE THRESHOLD statement.

For those event monitors that require data collection to be enabled, there are different options available to you. Depending on the type of event monitor you are working with, you might set a database configuration parameter to enable data collection across the entire database. Alternatively, you might choose to enable the collection of specific kinds of data for specific types of workload objects. For example, to configure the collection of basic information for a unit of work event monitor when any unit of work in the system finishes, you can set the `mon_uow_data` parameter to `BASE`. Alternatively, to capture unit of work information only for a specific workload, you can specify the `COLLECT UNIT OF WORK DATA BASE` clause as part of the `CREATE WORKLOAD` or `ALTER WORKLOAD` statements.

4. Run your applications or queries. After the event monitor has been created, and activated, and you have enabled data collection, run the applications or queries for which you want to collect data.
5. Optional: Deactivate the event monitor. After you run the applications or queries for which you want data collected, you can deactivate the event monitor using the `SET EVENT MONITOR STATE` statement. (see step 2 on page 32). Deactivating the event monitor is not necessary before proceeding to the next step, however leaving the event monitor active will result in disk space being used for data that you might not be interested in looking at.
6. Examine the data collected by the event monitor. Depending on the type of output the event monitor creates, there are different options for accessing the data collected. If the data is written directly to a relational table, you can use SQL to access the data contained in the table columns. On the other hand, if the event monitor writes to an unformatted event (UE) table, you must post-process the UE table using a command like `db2evmonfmt` or a procedure like `EVMON_FORMAT_UE_TO_TABLES` before you can view the event data.
7. Optional: Prune data that is no longer needed from the event monitor tables. For event monitors that you use on a regular basis, you might want to prune unneeded data from the tables. For example, if you use a unit of work event monitor to generate daily accounting reports about the system resources used by different applications, you might want to delete the current day's data from the event monitor tables once the reports have been generated.

**Tip:** If you need to prune event monitor output regularly, consider using an unformatted event (UE) table to record event monitor output. Starting in DB2 Version 10.1, UE tables can be pruned automatically after data is transferred to regular tables.

## Creating event monitors

You create different types of event monitors by using variations on the `CREATE EVENT MONITOR` statement. You can use the options for that statement to specify the type of data that event monitors collect and how the event monitors produce their output. The sections that follow describe the different output options and how to create event monitors that produce these types of output.

### Before you begin

Before creating an event monitor, it is important to understand the different options for the output that event monitors can produce. Most event monitors can produce output in at least two formats; some let you choose from up to four formats.

## Procedure

To create an event monitor:

1. Determine what kind of event monitor you need.
2. Decide what type of output you want from the event monitor. Do you want data to be written to a regular table, an unformatted event table, a file, or a pipe?
3. Issue a CREATE EVENT MONITOR statement.
4. Optional: If the type of event monitor that you created requires activation, activate it by issuing the SET EVENT MONITOR STATE statement.

## Output options for event monitors

Event monitors can report the data they collect in a number of ways. All event monitors can write the data they collect to tables; some write to unformatted event (UE) tables, which can help improve performance. Others can also write directly to a file or named pipe.

Depending on how you want to use the information collected by event monitors, and on the type of event monitor, you can choose to have the output that the event monitors collect produced in different ways. The output types available include:

### Regular tables

As of DB2 Version 10.1, all event monitors can write to regular tables that can be queried directly using SQL. For a given event, each of the monitor elements or metrics collected for the event is written to its own column in the table. This makes it possible to use a SELECT statement query the output to examine the values for a specific monitor element.

To create an event monitor that writes to tables, specify the WRITE TO TABLE clause in the CREATE EVENT MONITOR statement. Depending on the event monitor, one or more tables are created to contain the output, each table containing monitor elements that belong to a single logical group. See “Target tables, control tables, and event monitor table management” on page 88 for details about the specific table produced for each logical group.

Tables can be stored in a table space of your choosing; however the target table of a CREATE EVENT MONITOR statement must be a non-partitioned table.

**Note:** There are two types of event monitors that write to tables. The first type includes event monitors created in Version 9.7 and later releases. These include the unit of work, package cache, locking and change history event monitor. As of DB2 Version 10.1, the first three of these event monitors can write their output to regular tables as an alternative to UE tables. The change history event monitor writes only to regular tables.

The second type are the event monitors implemented before DB2 Version 9.7. These include all other event monitors.

Generally, after an event monitor of either type has been created, they work in much the same way. That is, you can use SQL to directly access the data in the tables that they produce. However, the older event monitors in the second category have additional options that you can specify when creating the event monitor. In addition, only event monitors in the second category are capable of writing also to files and named pipes.

### Unformatted event (UE) tables

UE tables were introduced in DB2 Version 9.7 for the new event monitors added in that release. UE tables are relational tables, however, they have only a limited number of columns. Most of the data associated with each event is written to a column containing an inline binary (BLOB) object. Writing event data in binary format reduces the time it takes to write each record to the table. For this reason, UE tables are particularly useful where event monitor performance is important, which might be the case on highly I/O or CPU-bound systems.

However, because the event data is written in binary format, you cannot use SQL to extract legible data. You must perform post-processing on the UE table to extract the data stored in binary format. Another benefit of using UE tables is that you can have UE table data pruned automatically during post-processing. The `EVMON_FORMAT_UE_TO_TABLES` procedure has an option to delete data from the UE table after it has been successfully extracted.

To create an event monitor that writes to an unformatted event table, specify the `WRITE TO UNFORMATTED EVENT TABLE` clause in the `CREATE EVENT MONITOR` statement. Only one UE table is created per event monitor.

**Files** Some event monitors support sending their output directly to files maintained by the file system. This type of output is useful if you do not want the event monitor output to be subject to the additional processing time caused when being managed within the database, or if you want to look at the data while the database is offline. To create an event monitor that writes to files, specify the `WRITE TO FILE` clause in the `CREATE EVENT MONITOR` statement.

### Named pipes

If you want to have an application process event data as it is generated, you can use a named pipe event monitor. These types of event monitors send their output directly to a named pipe so that the data can be used by another application immediately. This might be useful if you need to manipulate event data in real time.

To create an event monitor that writes to a named pipe, specify the `WRITE TO PIPE` clause in the `CREATE EVENT MONITOR` statement.

Depending on your needs, one type of event monitor output might be more appropriate than another. Table 5 provides an summary of when specific output types are particularly useful.

*Table 5. Summary of different event monitor output types*

| Output type                   | Scenarios where this output type is useful  |
|-------------------------------|---|
| Regular tables                | <ul style="list-style-type: none"><li>• When you want to examine monitoring data at a later point in time</li><li>• In systems that are not approaching the maximum capacity for CPU, log file or disk storage</li><li>• Where immediate access to data using SQL is desirable</li></ul>  |
| Unformatted event (UE) tables | <ul style="list-style-type: none"><li>• When you want to examine monitoring data at a later point in time</li><li>• In systems where event monitor performance is a priority, or where there are constraints on CPU, log file or disk usage</li><li>• Where the added step of post-processing of data is not an issue</li></ul> |

Table 5. Summary of different event monitor output types (continued)

| Output type | Scenarios where this output type is useful   |
|-------------|--|
| Files       | <ul style="list-style-type: none"> <li>• In systems where you do not want or need to manage monitor data as part of the database. (Eliminates the additional processing time of logging, inserts, maintaining consistency)</li> <li>• When you want to store the data outside of the database being monitored</li> <li>• When you want to examine the data offline at later point in time</li> </ul> |
| Pipes       | <ul style="list-style-type: none"> <li>• Streaming event data to an application that processes it immediately.</li> <li>• When there is no need to access event data at a later point in time.</li> </ul>  |

Not all event monitors support all output types. For example, only the unit of work, package cache and locking event monitor can produce a UE table. Table 6 shows what output options are available for different types of event monitors:

Table 6. Output options for event monitors

| Event monitor type          | Regular table | Unformatted event table | File | Named pipe |
|-----------------------------|---------------|-------------------------|------|------------|
| Activity                    | Yes           |                         | Yes  | Yes        |
| Buffer pool                 | Yes           |                         | Yes  | Yes        |
| Change history              | Yes           |                         |      |            |
| Connections                 | Yes           |                         | Yes  | Yes        |
| Database                    | Yes           |                         | Yes  | Yes        |
| Deadlocks* (all variations) | Yes           |                         | Yes  | Yes        |
| Locking                     | Yes           | Yes                     |      |            |
| Package cache               | Yes           | Yes                     |      |            |
| Statement                   | Yes           |                         | Yes  | Yes        |
| Statistics                  | Yes           |                         | Yes  | Yes        |
| Table space                 | Yes           |                         | Yes  | Yes        |
| Table                       | Yes           |                         | Yes  | Yes        |
| Threshold violations        | Yes           |                         | Yes  | Yes        |
| Transaction*                | Yes           |                         | Yes  | Yes        |
| Unit of work                | Yes           | Yes                     |      |            |
| * Deprecated event monitor. |               |                         |      |            |

### Event monitors that write to tables

Starting in DB2 Version 10.1, all event monitors can write output to regular tables that can be queried directly using SQL.

In addition, starting with DB2 Version 10.1, you can use the procedure `EVMON_UPGRADE_TABLES` to upgrade the tables produced by event monitors in earlier releases. This capability lets you more easily retain event monitor data as you upgrade your DB2 product.

### Creating event monitors that write to tables:

To create an event monitor, use the CREATE EVENT MONITOR STATEMENT. There are different forms of this statement that you use, depending on the type of events that you intend to monitor.

### Before you begin

- You need SQLADM or DBADM authority to create a table event monitor.
- The target table of a CREATE EVENT MONITOR statement - that is, the table to which the event monitor writes its output - must be a non-partitioned table.

### About this task

The various options for table event monitors are set in the CREATE EVENT MONITOR statement. For further assistance in generating CREATE EVENT MONITOR SQL statements for write-to-table event monitors, you can use the **db2evtb1** command. Simply provide the name of the event monitor and the required event type (or types), and the CREATE EVENT MONITOR statement is generated, complete with listings of all the target tables. You can then copy the generated statement, make modifications, and then execute the statement from the command line processor.

### Procedure

To create an event monitor that writes its output to a regular table, perform the following steps:

1. Formulate a CREATE EVENT MONITOR statement using the WRITE TO TABLE clause to indicate that event monitor data is to be collected in a table (or set of tables).

```
CREATE EVENT MONITOR evmon-name FOR eventtype
WRITE TO TABLE
```

Where evmon-name is the name of the event monitor, and eventtype is one of the following values:

- ACTIVITIES
- BUFFERPOOLS
- CHANGE HISTORY
- CONNECTIONS
- DATABASE
- DEADLOCKS
- LOCKING
- PACKAGE CACHE
- STATEMENTS
- STATISTICS
- TABLE
- TABLESPACE
- THRESHOLD VIOLATIONS
- TRANSACTIONS
- UNIT OF WORK

For example, to create a unit of work event monitor called myevmon, use a statement like the one that follows:

```
CREATE EVENT MONITOR myevmon FOR UNIT OF WORK
WRITE TO TABLE
```

The preceding statement creates a unit of work event monitor that uses defaults for the logical groups of monitor elements collected, the corresponding output

table names, and the target table spaces for the tables. For more information on these defaults, refer to the documentation for the appropriate CREATE EVENT MONITOR statement.

- Optional: Specify the logical groups for which you want data collected. By default, event data is collected for all logical data groups for the event monitor type. (See “Target tables, control tables, and event monitor table management” on page 88 for details.) If you want only data for selected logical groups collected, you can specify the names of the logical groups to include in the CREATE EVENT MONITOR statement. For example, with a locking event monitor, you might want to collect only the information associated with the LOCK and PARTICIPANT logical groups. To include only these logical groups, you can use a statement like the one that follows:

```
CREATE EVENT MONITOR mylocks FOR LOCKING
WRITE TO TABLE
LOCK, PARTICIPANTS
```

- Optional: Specify the table names to use for the output tables. Unless you specify otherwise, default names are used for the tables for each logical group of monitor elements. The default name used is derived by concatenating the logical group name with the name of the event monitor. For example, for the locking event monitor created by the statement in the preceding step, the unqualified names for the tables produced are LOCK\_MYLOCKS and PARTICIPANTS\_MYLOCKS. To override the default names, include the table names to use when specifying the logical groups:

```
CREATE EVENT MONITOR mylocks FOR LOCKING
WRITE TO TABLE
LOCK(TABLE LOCKDATA), PARTICIPANTS(TABLE PARTICIP)
```

In the preceding example, the names used for the tables for the LOCK and PARTICIPANTS logical groups are LOCKDATA\_MYLOCKS and PARTICIP\_MYLOCKS.

You can also override the table space to be used for each table by including the name of the table space to use:

```
CREATE EVENT MONITOR mylocks FOR LOCKING
WRITE TO TABLE
LOCK(TABLE LOCKDATA IN EVMONSPACE), PARTICIPANTS(TABLE PARTICIP IN EVMONSPACE)
```

In the preceding example, the EVMONSPACE table space is used for both output tables.

### Additional options

Different event monitors provide different configuration options. For details on the options available for a specific type of event monitor, refer to the documentation for the CREATE EVENT MONITOR statement for the type of event monitor you want to use. The examples that follow show some of the configuration options you can choose for different event monitors:

#### Capturing multiple event types with a single event monitor

Some types<sup>2</sup> of event monitors can capture different types of events with a single event monitor. If you want to capture multiple types of events with this event monitor, specify additional values for eventtype, separated by a comma. For example, you might want to combine bufferpool and table space monitoring in a single event monitor:

---

2. Event monitors for BUFFERPOOLS, CONNECTIONS, DATABASE, DEADLOCKS, STATEMENTS, TABLES, and TABLESPACES support this option.

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
WRITE TO TABLE
```

This event monitor will monitor for the BUFFERPOOL and TABLESPACE event types. Assuming that the previously listed statement was issued by the user dbadmin, the derived names and table spaces of the target tables are as follows:

- DBADMIN.BUFFERPOOL\_MYEVMON
- DBADMIN.TABLESPACE\_MYEVMON
- DBADMIN.CONTROL\_,MYEVMON

### Adjusting the size of event monitor output buffers

You can alter the size of the table event monitor buffers (in 4K pages) for some types<sup>2</sup> of event monitors by adjusting the BUFFERSIZE value. For example, in the following statement:

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
WRITE TO TABLE BUFFERSIZE 8
```

8 is the combined capacity (in 4K pages) of the two event table buffers. This adds up to 32K of buffer space; 16K for each buffer.

The default size of each buffer is 4 pages (two 16K buffers are allocated). The minimum size is 1 page. The maximum size of the buffers is limited by the size of the monitor heap, because the buffers are allocated from that heap. For performance reasons, highly active event monitors should have larger buffers than relatively inactive event monitors.

### Controlling whether event monitor output is blocked or non-blocked

Some event monitors<sup>2</sup> let you control how to proceed when event monitor output buffers are full. For blocked event monitors, each agent that generates an event will wait for the event buffers to be written to table if they are full. This can degrade database performance, as the suspended agent and any dependent agents cannot run until the buffers are clear. Use the BLOCKED clause to ensure no losses of event data:

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
WRITE TO TABLE BUFFERSIZE 8 BLOCKED
```

If database performance is of greater importance than collecting every single event record, use non-blocked event monitors. In this case, each agent that generates an event will not wait for the event buffers to be written to table if they are full. As a result, non-blocked event monitors are subject to data loss on highly active systems. Use the NONBLOCKED clause to minimize the additional processing time caused by event monitoring:

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
```

**Note:** See “Target tables, control tables, and event monitor table management” on page 88 and “Write-to-table and file event monitor buffering” on page 107 for additional information about how information about discarded events is written to the control table for the event monitor.

### Controlling what monitor elements for which data is collected

Which monitor elements to collect data for. If you are interested in only a few monitor elements, you can specify which ones you want to collect for some event monitors<sup>2</sup> by specifying the element name in the CREATE EVENT MONITOR statement:

```

CREATE EVENT MONITOR myevmon FOR DATABASE, BUFFERPOOLS, TABLESPACES
    WRITE TO TABLE DB, DBMEMUSE,
    BUFFERPOOL (EXCLUDES(db_path, files_closed)),
    TABLESPACE (INCLUDES
        (tablespace_name, direct_reads, direct_writes))
    BUFFERSIZE 8 NONBLOCKED

```

All the monitor elements for the DB and DBMEMUSE logical data groups are captured (this is the default behavior). For BUFFERPOOL, all monitor elements except **db\_path** and **files\_closed** are captured. And finally, for TABLESPACE, **tablespace\_name**, **direct\_reads** and **direct\_writes** are the only monitor elements captured.

### Setting a threshold for deactivating an event monitor based on table space used

All event monitors provide the option to specify how full the table space can get before the event monitor automatically deactivates:

```

CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
    PCTDEACTIVATE 90

```

When the table space reaches 90% capacity, the myevmon event monitor automatically shuts off. The PCTDEACTIVATE clause can only be used for DMS table spaces. If the target table space has auto-resize enabled, set the PCTDEACTIVATE clause to 100.

### What to do next

By default, event monitors that were introduced in Version 9.7 or later are created as AUTOSTART event monitors. They are activated automatically when the database is next activated, and on subsequent database activations thereafter. If you want to activate the event monitor immediately, before the next database activation, use the SET EVENT MONITOR STATE statement to manually start the event monitor. In addition for each of the locking, unit of work and package cache event monitors, you must also enable data collection.

### Event monitor logical data groups and monitor elements:

Monitor elements that are often useful to examine together are grouped in *logical data groups*.

All event monitors use logical data groups in one way or another. For some event monitor types, you can specify what information you want to collect by specifying the logical data groups for which you want information recorded. Logical data groups are also used to group data together in the output event monitors generate; for example, event monitors that write to tables generally create one table for each logical data group of monitor elements.

The following table lists the logical data groupings and monitor elements that can be returned by event monitoring.

- “changesummary logical data group” on page 42
- “dbdbmcfg logical data group” on page 42
- “ddlstmtexec logical data group” on page 42
- “dllock logical data group” on page 43
- “event\_activity logical data group” on page 43
- “event\_activitymetrics logical data group” on page 45
- “event\_activitystmt logical data group” on page 48
- “event\_activityvals logical data group” on page 49

- “event\_bufferpool logical data group” on page 49
- “event\_conn logical data group” on page 51
- “event\_connheader logical data group” on page 53
- “event\_connmemuse logical data group” on page 54
- “event\_data\_value logical data group” on page 54
- “event\_db logical data group” on page 54
- “event\_dbheader logical data group” on page 58
- “event\_dbmemuse logical data group” on page 58
- “event\_deadlock logical data group” on page 59
- “event\_detailed\_dlconn logical data group” on page 59
- “event\_dlconn logical data group” on page 60
- “event\_histogrambin logical data group” on page 61
- “event\_log\_header logical data group” on page 61
- “event\_overflow logical data group” on page 61
- “event\_qstats logical data group” on page 61
- “event\_scstats logical data group” on page 62
- “event\_start logical data group” on page 63
- “event\_stmt logical data group” on page 63
- “event\_stmt\_history logical data group” on page 65
- “event\_subsection logical data group” on page 65
- “event\_table logical data group” on page 66
- “event\_tablespace logical data group” on page 66
- “event\_thresholdviolations logical data group” on page 68
- “event\_wlstats logical data group” on page 69
- “event\_wcstats logical data group” on page 68
- “event\_xact logical data group” on page 70
- “evmonstart logical data group” on page 71
- “lock logical data group” on page 71
- “lock\_participants logical data group” on page 72
- “lock\_participant\_activities logical data group” on page 71
- “lock\_activity\_values logical data group” on page 71
- “pkgcache logical data group” on page 74
- “pkgcache\_metrics logical data group” on page 75
- “pkgcache\_stmt\_args logical data group” on page 79
- “regvar logical data group” on page 79
- “sqlca logical data group” on page 79
- “txncompletion logical data group” on page 79
- “uow logical data group” on page 80
- “uow\_metrics logical data group” on page 81
- “uow\_package\_list logical data group” on page 87
- “uow\_executable\_list logical data group” on page 81
- “utillocation logical data group” on page 87
- “utilphase logical data group” on page 88
- “utilstart logical data group” on page 88
- “utilstop logical data group” on page 88

### **changesummary logical data group**

- "event\_id - Event ID monitor element" on page 810
- "event\_type - Event Type monitor element" on page 812
- "event\_timestamp - Event timestamp monitor element" on page 811
- "member - Database member monitor element" on page 957
- "coord\_member - Coordinator member monitor element" on page 749
- "utility\_invocation\_id - Utility invocation ID" on page 1413
- "utility\_type - Utility Type" on page 1417
- "appl\_id - Application ID monitor element" on page 666
- "appl\_name - Application name monitor element" on page 670
- "application\_handle - Application handle monitor element" on page 675
- "system\_auth\_id - System authorization identifier monitor element" on page 1267
- "session\_auth\_id - Session authorization ID monitor element" on page 1211
- "client\_platform - Client operating platform monitor element" on page 714
- "client\_protocol - Client communication protocol monitor element" on page 716
- "client\_port\_number - Client port number monitor element" on page 715
- "client\_pid - Client process ID monitor element" on page 713
- "client\_hostname - Client hostname monitor element" on page 711
- "client\_wrkstnname - Client workstation name monitor element" on page 717
- "client\_acctng - Client accounting string monitor element" on page 709
- "client\_userid - Client user ID monitor element" on page 717
- "client\_applname - Client application name monitor element" on page 710
- "backup\_timestamp - Backup timestamp" on page 689

### **dbdbmcfg logical data group**

- "event\_id - Event ID monitor element" on page 810
- "event\_timestamp - Event timestamp monitor element" on page 811
- "member - Database member monitor element" on page 957
- "event\_type - Event Type monitor element" on page 812
- "cfg\_name - Configuration name" on page 704
- "cfg\_value - Configuration value" on page 706
- "cfg\_value\_flags - Configuration value flags" on page 706
- "cfg\_old\_value - Configuration old value" on page 705
- "cfg\_old\_value\_flags - Configuration old value flags" on page 705
- "cfg\_collection\_type - Configuration collection type" on page 704
- "deferred - Deferred" on page 785

### **ddlstmexec logical data group**

- "event\_id - Event ID monitor element" on page 810
- "event\_timestamp - Event timestamp monitor element" on page 811
- "member - Database member monitor element" on page 957
- "event\_type - Event Type monitor element" on page 812
- "global\_transaction\_id - Global transaction identifier monitor element" on page 851

"local\_transaction\_id - Local transaction identifier monitor element" on page 900  
"savepoint\_id - Savepoint ID" on page 1198  
"uow\_id - Unit of work ID monitor element" on page 1401  
"ddl\_classification - DDL classification" on page 780  
"stmt\_text - SQL statement text monitor element" on page 1255

#### **dllock logical data group**

"data\_partition\_id - Data partition identifier monitor element" on page 767  
"lock\_attributes - Lock attributes monitor element" on page 901  
"lock\_count - Lock count monitor element" on page 902  
"lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
"lock\_escalation - Lock escalation monitor element" on page 904  
"lock\_hold\_count - Lock hold count monitor element" on page 911  
"lock\_mode - Lock mode monitor element" on page 912  
"lock\_name - Lock name monitor element" on page 914  
"lock\_object\_name - Lock Object Name" on page 915  
"lock\_object\_type - Lock object type waited on monitor element" on page 916  
"lock\_release\_flags - Lock release flags monitor element" on page 918  
"lock\_status - Lock status monitor element" on page 919  
"node\_number - Node Number" on page 970  
"table\_file\_id - Table file ID monitor element" on page 1268  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"tablespace\_name - Table space name monitor element" on page 1279

**Note:** The underlying implementation for this logical data group is the snapshot monitor LOCK logical data group. If you examine the output for this logical group in the self-describing stream used for the file and pipe output options, you can see that the LOCK group is used to generate the output.

#### **event\_activity logical data group**

"act\_exec\_time - Activity execution time monitor element" on page 639  
"activate\_timestamp - Activate timestamp monitor element" on page 644  
"activity\_id - Activity ID monitor element" on page 645  
"activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
"activity\_type - Activity type monitor element" on page 647  
"address - IP address from which the connection was initiated" on page 649  
"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_name - Application name monitor element" on page 670  
"arm\_correlator - Application response measurement correlator monitor element" on page 677  
"coord\_partition\_num - Coordinator partition number monitor element" on page 750  
"db\_work\_action\_set\_id - Database work action set ID monitor element" on page 777

"db\_work\_class\_id - Database work class ID monitor element" on page 778  
 details\_xml (This XML document is a metrics document of type activity\_metrics, as described in the XML schema document sql1lib/misc/DB2MonCommon.xsd.) You can also access the metrics reported in this document through the event\_activitymetrics logical data group.  
 "intra\_parallel\_state - Current state of intrapartition parallelism monitor element" on page 886  
 "num\_remaps - Number of remaps monitor element" on page 981  
 "parent\_activity\_id - Parent activity ID monitor element" on page 1013  
 "parent\_uow\_id - Parent unit of work ID monitor element" on page 1014  
 "partial\_record - Partial Record monitor element" on page 1014  
 "pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048  
 "pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050  
 "query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element" on page 1162  
 "sc\_work\_action\_set\_id - Service class work action set ID monitor element" on page 1198  
 "sc\_work\_class\_id - Service class work class ID monitor element" on page 1199  
 "section\_actuals - Section actuals monitor element" on page 1200  
 "service\_subclass\_name - Service subclass name monitor element" on page 1208  
 "service\_superclass\_name - Service superclass name monitor element" on page 1209  
 "session\_auth\_id - Session authorization ID monitor element" on page 1211  
 "sort\_overflows - Sort overflows monitor element" on page 1224  
 "sqlca - SQL Communications Area (SQLCA)" on page 1229  
 "time\_completed - Time completed monitor element" on page 1308  
 "time\_created - Time created monitor element" on page 1309  
 "time\_started - Time started monitor element" on page 1310  
 "total\_sort\_time - Total sort time monitor element" on page 1374  
 "total\_sorts - Total sorts monitor element" on page 1375  
 "tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
 "tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
 "tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
 "tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389  
 "uow\_id - Unit of work ID monitor element" on page 1401  
 "workload\_id - Workload ID monitor element" on page 1426  
 "workload\_occurrence\_id - Workload occurrence identifier monitor element" on page 1428  
 "pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080  
 "pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116  
 “pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118  
 “pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120  
 “pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121  
 “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123  
 “pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125  
 “pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136  
 “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139  
 “prep\_time - Preparation time monitor element” on page 1154  
 “query\_card\_estimate - Query Number of Rows Estimate” on page 1163  
 “query\_cost\_estimate - Query cost estimate monitor element” on page 1163  
 “rows\_fetched - Rows fetched monitor element” on page 1188  
 “rows\_modified - Rows modified monitor element” on page 1190  
 “rows\_returned - Rows returned monitor element” on page 1193  
 “system\_cpu\_time - System CPU time monitor element” on page 1267  
 “user\_cpu\_time - User CPU time monitor element” on page 1411  
 “wl\_work\_action\_set\_id - Workload work action set identifier monitor element” on page 1420  
 “wl\_work\_class\_id - Workload work class identifier monitor element” on page 1421  
 “mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
 “member - Database member monitor element” on page 957  
 “query\_data\_tag\_list - Estimated query data tag list monitor element” on page 1164  
 “total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378  
 “total\_stats\_fabrications - Total statistics fabrications monitor elements” on page 1379  
 “total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element” on page 1382  
 “total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380

#### **event\_activitymetrics logical data group**

“audit\_events\_total - Total audit events monitor element” on page 679  
 “audit\_file\_writes\_total - Total audit files written monitor element” on page 682  
 “audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683  
 “audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element” on page 685

"coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element" on page 751

"deadlocks - Deadlocks detected monitor element" on page 783

"diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element" on page 788

"diaglog\_writes\_total - Total diagnostic log file writes monitor element" on page 789

"direct\_read\_reqs - Direct read requests monitor element" on page 790

"direct\_read\_time - Direct read time monitor element" on page 792

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_write\_reqs - Direct write requests monitor element" on page 796

"direct\_write\_time - Direct write time monitor element" on page 798

"direct\_writes - Direct writes to database monitor element" on page 800

"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823

"fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element" on page 824

"fcm\_message\_rcvs\_total - Total FCM message receives monitor element" on page 826

"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827

"fcm\_message\_send\_wait\_time - FCM message send wait time monitor element" on page 828

"fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830

"fcm\_rcv\_volume - FCM received volume monitor element" on page 831

"fcm\_rcv\_wait\_time - FCM received wait time monitor element" on page 832

"fcm\_rcvs\_total - FCM receives total monitor element" on page 834

"fcm\_send\_volume - FCM send volume monitor element" on page 835

"fcm\_send\_wait\_time - FCM send wait time monitor element" on page 836

"fcm\_sends\_total - FCM sends total monitor element" on page 838

"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839

"fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element" on page 840

"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842

"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843

"fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element" on page 844

"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846

"lock\_escals - Number of lock escalations monitor element" on page 905

"lock\_timeouts - Number of lock timeouts monitor element" on page 920

"lock\_wait\_time - Time waited on locks monitor element" on page 924

"lock\_waits - Lock waits monitor element" on page 928

"log\_buffer\_wait\_time - Log buffer wait time monitor element" on page 933

“log\_disk\_wait\_time - Log disk wait time monitor element” on page 935

“log\_disk\_waits\_total - Total log disk waits monitor element” on page 936

“num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element” on page 975

“num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element” on page 979

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144

“post\_threshold\_sorts - Post threshold sorts monitor element” on page 1150

“rows\_modified - Rows modified monitor element” on page 1190

“rows\_read - Rows read monitor element” on page 1191

“rows\_returned - Rows returned monitor element” on page 1193

“sort\_overflows - Sort overflows monitor element” on page 1224

“stmt\_exec\_time - Statement execution time monitor element” on page 1245

“thresh\_violations - Number of threshold violations monitor element” on page 1304

“total\_act\_time - Total activity time monitor element” on page 1311

"total\_act\_wait\_time - Total activity wait time monitor element" on page 1312  
 "total\_app\_section\_executions - Total application section executions monitor element" on page 1317  
 "total\_cpu\_time - Total CPU time monitor element" on page 1330  
 "total\_routine\_invocations - Total routine invocations monitor elements" on page 1355  
 "total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element" on page 1356  
 "total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements" on page 1356  
 "total\_routine\_time - Total routine time monitor element" on page 1357  
 "total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element" on page 1359  
 "total\_routine\_user\_code\_time - Total routine user code time monitor element" on page 1360  
 "total\_section\_proc\_time - Total section processing time monitor element" on page 1367  
 "total\_section\_sort\_proc\_time - Total section sort processing time monitor element" on page 1368  
 "total\_section\_sort\_time - Total section sort time monitor element" on page 1370  
 "total\_section\_sorts - Total section sorts monitor element" on page 1371  
 "total\_section\_time - Total section time monitor element" on page 1373  
 "total\_sorts - Total sorts monitor element" on page 1375  
 "tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396  
 "wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element" on page 1421  
 "wlm\_queue\_time\_total - Workload manager total queue time monitor element" on page 1422

#### **event\_activitystmt logical data group**

"activate\_timestamp - Activate timestamp monitor element" on page 644  
 "activity\_id - Activity ID monitor element" on page 645  
 "activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
 "appl\_id - Application ID monitor element" on page 666  
 "comp\_env\_desc - Compilation environment monitor element" on page 722  
 "creator - Application Creator" on page 763  
 "eff\_stmt\_text - Effective statement text monitor element" on page 805  
 "executable\_id - Executable ID monitor element" on page 818  
 "intra\_parallel\_state - Current state of intrapartition parallelism monitor element" on page 886  
 "package\_name - Package name monitor element" on page 1006  
 "package\_version\_id - Package version monitor element" on page 1007  
 "query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element" on page 1162  
 "routine\_id - Routine ID monitor element" on page 1187  
 "section\_env - Section environment monitor element" on page 1201  
 "section\_number - Section number monitor element" on page 1201

"stmt\_first\_use\_time - Statement first use timestamp monitor element" on page 1246  
"stmt\_invocation\_id - Statement invocation identifier monitor element" on page 1247  
"stmt\_isolation - Statement isolation" on page 1248  
"stmt\_last\_use\_time - Statement last use timestamp monitor element" on page 1248  
"stmt\_lock\_timeout - Statement lock timeout monitor element" on page 1249  
"stmt\_nest\_level - Statement nesting level monitor element" on page 1249  
"stmt\_pkgcache\_id - Statement package cache identifier monitor element" on page 1251  
"stmt\_query\_id - Statement query identifier monitor element" on page 1252  
"stmt\_source\_id - Statement source identifier" on page 1253  
"stmt\_text - SQL statement text monitor element" on page 1255  
"stmt\_type - Statement type monitor element" on page 1256  
"uow\_id - Unit of work ID monitor element" on page 1401  
"member - Database member monitor element" on page 957

#### **event\_activityvals logical data group**

"activate\_timestamp - Activate timestamp monitor element" on page 644  
"activity\_id - Activity ID monitor element" on page 645  
"activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
"appl\_id - Application ID monitor element" on page 666  
"intra\_parallel\_state - Current state of intrapartition parallelism monitor element" on page 886  
"query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element" on page 1162  
"stmt\_value\_data - Value data" on page 1259  
"stmt\_value\_index - Value index" on page 1259  
"stmt\_value\_isnull - Value has null value monitor element" on page 1260  
"stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261  
"stmt\_value\_type - Value type monitor element" on page 1261  
"uow\_id - Unit of work ID monitor element" on page 1401  
"member - Database member monitor element" on page 957

#### **event\_bufferpool logical data group**

"bp\_id - Buffer pool identifier monitor element" on page 693  
"bp\_name - Buffer pool name monitor element" on page 693  
"db\_name - Database name monitor element" on page 773  
"db\_path - Database Path" on page 774  
"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_read\_time - Direct read time monitor element" on page 792  
"direct\_reads - Direct reads from database monitor element" on page 794  
"direct\_write\_reqs - Direct write requests monitor element" on page 796  
"direct\_write\_time - Direct write time monitor element" on page 798  
"direct\_writes - Direct writes to database monitor element" on page 800

“event\_time - Event Time” on page 811

“evmon\_activates - Number of event monitor activations” on page 813

“evmon\_flushes - Number of Event Monitor Flushes” on page 819

“files\_closed - Database files closed monitor element” on page 848

“partial\_record - Partial Record monitor element” on page 1014

“pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element” on page 1026

“pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element” on page 1027

“pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element” on page 1028

“pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element” on page 1032

“pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element” on page 1033

“pool\_async\_read\_time - Buffer Pool Asynchronous Read Time” on page 1033

“pool\_async\_write\_time - Buffer pool asynchronous write time monitor element” on page 1034

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“block\_ios - Number of block I/O requests monitor element” on page 690

“pages\_from\_block\_ios - Total number of pages read by block I/O monitor element” on page 1011

“pages\_from\_vectored\_ios - Total number of pages read by vectored I/O monitor element” on page 1012

“pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element” on page 1031

“pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element” on page 1038

“pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element” on page 1038

“pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element” on page 1039

“pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element” on page 1087

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139  
“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141  
“unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399  
“vectored\_ios - Number of vectored I/O requests monitor element” on page 1419

#### **event\_conn logical data group**

“acc\_curs\_blk - Accepted Block Cursor Requests” on page 636  
“agent\_id - Application handle (agent ID) monitor element” on page 650  
“appl\_id - Application ID monitor element” on page 666  
“appl\_priority - Application Agent Priority” on page 671  
“appl\_priority\_type - Application Priority Type” on page 672  
“appl\_section\_inserts - Section Inserts monitor element” on page 672  
“appl\_section\_lookups - Section Lookups” on page 673  
“authority\_bitmap - User authorization level monitor element” on page 687  
“authority\_lvl - User authorization level monitor element” on page 688  
“binds\_precompiles - Binds/Precompiles Attempted” on page 690  
“cat\_cache\_inserts - Catalog cache inserts monitor element” on page 698  
“cat\_cache\_lookups - Catalog cache lookups monitor element” on page 699  
“cat\_cache\_overflows - Catalog Cache Overflows” on page 700  
“commit\_sql\_stmts - Commit Statements Attempted” on page 721  
“ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements” on page 781  
“deadlocks - Deadlocks detected monitor element” on page 783  
“direct\_read\_reqs - Direct read requests monitor element” on page 790  
“direct\_read\_time - Direct read time monitor element” on page 792  
“direct\_reads - Direct reads from database monitor element” on page 794  
“direct\_write\_reqs - Direct write requests monitor element” on page 796  
“direct\_write\_time - Direct write time monitor element” on page 798  
“direct\_writes - Direct writes to database monitor element” on page 800  
“disconn\_time - Database Deactivation Timestamp” on page 803  
“dynamic\_sql\_stmts - Dynamic SQL Statements Attempted” on page 804  
“failed\_sql\_stmts - Failed Statement Operations” on page 821  
“hash\_join\_overflows - Hash Join Overflows” on page 867  
“hash\_join\_small\_overflows - Hash Join Small Overflows” on page 867  
“int\_auto\_rebinds - Internal Automatic Rebinds” on page 879  
“int\_commits - Internal commits monitor element” on page 880  
“int\_deadlock\_rollback - Internal Rollbacks Due To Deadlock” on page 881  
“int\_rollback - Internal rollbacks monitor element” on page 882  
“int\_rows\_deleted - Internal Rows Deleted” on page 884  
“int\_rows\_inserted - Internal Rows Inserted” on page 884  
“int\_rows\_updated - Internal Rows Updated” on page 885  
“lock\_escalation - Lock escalation monitor element” on page 904  
“lock\_timeouts - Number of lock timeouts monitor element” on page 920  
“lock\_wait\_time - Time waited on locks monitor element” on page 924

"lock\_waits - Lock waits monitor element" on page 928

"olap\_func\_overflows - OLAP Function Overflows monitor element" on page 997

"partial\_record - Partial Record monitor element" on page 1014

"int\_rows\_updated - Internal Rows Updated" on page 885

"pkg\_cache\_inserts - Package cache inserts monitor element" on page 1020

"pkg\_cache\_lookups - Package cache lookups monitor element" on page 1021

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121

"pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128

"prefetch\_wait\_time - Time waited for prefetch monitor element" on page 1151

"priv\_workspace\_num\_overflows - Private Workspace Overflows" on page 1156

"priv\_workspace\_section\_inserts - Private Workspace Section Inserts" on page 1157

"priv\_workspace\_section\_lookups - Private Workspace Section Lookups" on page 1158

"priv\_workspace\_size\_top - Maximum Private Workspace Size" on page 1158

"rej\_curs\_blk - Rejected Block Cursor Requests" on page 1173

"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184

"rows\_read - Rows read monitor element" on page 1191

"rows\_selected - Rows Selected" on page 1195

"rows\_written - Rows Written" on page 1196

"select\_sql\_stmts - Select SQL Statements Executed" on page 1203

"sequence\_no - Sequence number monitor element" on page 1204

"shr\_workspace\_num\_overflows - Shared Workspace Overflows" on page 1211

"shr\_workspace\_section\_inserts - Shared Workspace Section Inserts" on page 1212

"shr\_workspace\_section\_lookups - Shared Workspace Section Lookups" on page 1213

“shr\_workspace\_size\_top - Maximum Shared Workspace Size” on page 1213  
 “sort\_overflows - Sort overflows monitor element” on page 1224  
 “static\_sql\_stmts - Static SQL Statements Attempted” on page 1241  
 “system\_cpu\_time - System CPU time monitor element” on page 1267  
 “total\_hash\_joins - Total Hash Joins” on page 1338  
 “total\_hash\_loops - Total Hash Loops” on page 1338  
 “total\_olap\_funcs - Total OLAP Functions monitor element” on page 1346  
 “total\_sec\_cons - Secondary Connections” on page 1366  
 “total\_sort\_time - Total sort time monitor element” on page 1374  
 “total\_sorts - Total sorts monitor element” on page 1375  
 “uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed” on page 1398  
 “unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399  
 “user\_cpu\_time - User CPU time monitor element” on page 1411  
 “x\_lock\_escals - Exclusive lock escalations monitor element” on page 1429  
 “xquery\_stmts - XQuery Statements Attempted” on page 1431  
 “appl\_status - Application status monitor element” on page 673  
 “cat\_cache\_size\_top - Catalog cache high watermark monitor element” on page 701  
 “coord\_node - Coordinating Node” on page 750  
 “elapsed\_exec\_time - Statement Execution Elapsed Time” on page 806  
 “evmon\_flushes - Number of Event Monitor Flushes” on page 819  
 “lock\_escals - Number of lock escalations monitor element” on page 905  
 “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123  
 “pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125  
 “pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136  
 “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139  
 “pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141  
 “rows\_deleted - Rows deleted monitor element” on page 1188  
 “rows\_inserted - Rows inserted monitor element” on page 1189  
 “rows\_updated - Rows updated monitor element” on page 1196  
 CAT\_CACHE\_HEAP\_FULL

#### **event\_connheader logical data group**

“agent\_id - Application handle (agent ID) monitor element” on page 650  
 “appl\_id - Application ID monitor element” on page 666  
 “appl\_name - Application name monitor element” on page 670  
 “auth\_id - Authorization ID” on page 686  
 “client\_db\_alias - Database Alias Used by Application” on page 711  
 “client\_pid - Client process ID monitor element” on page 713  
 “client\_platform - Client operating platform monitor element” on page 714  
 “client\_prdid - Client product and version ID monitor element” on page 715

"client\_protocol - Client communication protocol monitor element" on page 716  
"codepage\_id - ID of Code Page Used by Application" on page 718  
"conn\_time - Time of database connection monitor element" on page 736  
"corr\_token - DRDA Correlation Token" on page 751  
"execution\_id - User Login ID" on page 820  
"node\_number - Node Number" on page 970  
"sequence\_no - Sequence number monitor element" on page 1204  
"territory\_code - Database Territory Code" on page 1303  
"client\_nname - Client name monitor element" on page 713

#### **event\_connmemuse logical data group**

"node\_number - Node Number" on page 970  
"pool\_config\_size - Configured Size of Memory Pool" on page 1040  
"pool\_cur\_size - Current Size of Memory Pool" on page 1041  
"pool\_id - Memory Pool Identifier" on page 1071  
"pool\_secondary\_id - Memory Pool Secondary Identifier" on page 1114  
"pool\_watermark - Memory Pool Watermark" on page 1127  
"appl\_id - Application ID monitor element" on page 666  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819  
POOL\_LIST\_ID  
POOL\_MAX\_SIZE

#### **event\_data\_value logical data group**

"deadlock\_id - Deadlock Event Identifier" on page 782  
"deadlock\_node - Partition Number Where Deadlock Occurred" on page 783  
"evmon\_activates - Number of event monitor activations" on page 813  
"participant\_no - Participant within Deadlock" on page 1015  
"stmt\_history\_id - Statement history identifier" on page 1246  
"stmt\_value\_data - Value data" on page 1259  
"stmt\_value\_index - Value index" on page 1259  
"stmt\_value\_isnull - Value has null value monitor element" on page 1260  
"stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261  
"stmt\_value\_type - Value type monitor element" on page 1261

#### **event\_db logical data group**

"active\_hash\_joins - Active hash joins" on page 644  
"appl\_section\_inserts - Section Inserts monitor element" on page 672  
"appl\_section\_lookups - Section Lookups" on page 673  
"async\_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 678  
"binds\_precompiles - Binds/Precompiles Attempted" on page 690  
"blocks\_pending\_cleanup - Pending cleanup rolled-out blocks monitor element" on page 692  
"cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698  
"cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699  
"cat\_cache\_overflows - Catalog Cache Overflows" on page 700

"cat\_cache\_size\_top - Catalog cache high watermark monitor element" on page 701  
 "catalog\_node - Catalog Node Number" on page 702  
 "catalog\_node\_name - Catalog Node Network Name" on page 702  
 "commit\_sql\_stmts - Commit Statements Attempted" on page 721  
 "connections\_top - Maximum Number of Concurrent Connections" on page 737  
 "db\_heap\_top - Maximum Database Heap Allocated" on page 773  
 "ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements" on page 781  
 "deadlocks - Deadlocks detected monitor element" on page 783  
 "direct\_read\_reqs - Direct read requests monitor element" on page 790  
 "direct\_read\_time - Direct read time monitor element" on page 792  
 "direct\_reads - Direct reads from database monitor element" on page 794  
 "direct\_write\_reqs - Direct write requests monitor element" on page 796  
 "direct\_write\_time - Direct write time monitor element" on page 798  
 "direct\_writes - Direct writes to database monitor element" on page 800  
 "disconn\_time - Database Deactivation Timestamp" on page 803  
 "dynamic\_sql\_stmts - Dynamic SQL Statements Attempted" on page 804  
 "evmon\_activates - Number of event monitor activations" on page 813  
 "evmon\_flushes - Number of Event Monitor Flushes" on page 819  
 "failed\_sql\_stmts - Failed Statement Operations" on page 821  
 "files\_closed - Database files closed monitor element" on page 848  
 "hash\_join\_overflows - Hash Join Overflows" on page 867  
 "hash\_join\_small\_overflows - Hash Join Small Overflows" on page 867  
 "int\_auto\_rebinds - Internal Automatic Rebinds" on page 879  
 "int\_commits - Internal commits monitor element" on page 880  
 "int\_rollbacks - Internal rollbacks monitor element" on page 882  
 "int\_rows\_deleted - Internal Rows Deleted" on page 884  
 "int\_rows\_inserted - Internal Rows Inserted" on page 884  
 "int\_rows\_updated - Internal Rows Updated" on page 885  
 "lock\_escals - Number of lock escalations monitor element" on page 905  
 "lock\_timeouts - Number of lock timeouts monitor element" on page 920  
 "lock\_wait\_time - Time waited on locks monitor element" on page 924  
 "lock\_waits - Lock waits monitor element" on page 928  
 "log\_held\_by\_dirty\_pages - Amount of Log Space Accounted for by Dirty Pages" on page 937  
 "log\_read\_time - Log Read Time" on page 938  
 "log\_reads - Number of Log Pages Read" on page 938  
 "log\_to\_redo\_for\_recovery - Amount of Log to be Redone for Recovery" on page 939  
 "log\_write\_time - Log Write Time" on page 940  
 "log\_writes - Number of Log Pages Written" on page 940  
 "num\_log\_read\_io - Number of Log Reads" on page 978  
 "num\_log\_write\_io - Number of Log Writes" on page 978  
 "num\_threshold\_violations - Number of threshold violations monitor element" on page 981

"olap\_func\_overflows - OLAP Function Overflows monitor element" on page 997

"partial\_record - Partial Record monitor element" on page 1014

"pkg\_cache\_inserts - Package cache inserts monitor element" on page 1020

"pkg\_cache\_lookups - Package cache lookups monitor element" on page 1021

"pkg\_cache\_num\_overflows - Package Cache Overflows" on page 1023

"pkg\_cache\_size\_top - Package cache high watermark" on page 1023

"pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element" on page 1026

"pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element" on page 1027

"pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element" on page 1028

"pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element" on page 1031

"pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element" on page 1032

"pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element" on page 1033

"pool\_async\_read\_time - Buffer Pool Asynchronous Read Time" on page 1033

"pool\_async\_write\_time - Buffer pool asynchronous write time monitor element" on page 1034

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_drty\_pg\_steal\_clns - Buffer pool victim page cleaners triggered monitor element" on page 1055

"pool\_drty\_pg\_thrsh\_clns - Buffer pool threshold cleaners triggered monitor element" on page 1056

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"pool\_lsn\_gap\_clns - Buffer pool log space cleaners triggered monitor element" on page 1086

"pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element" on page 1087

"pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“post\_shrthreshold\_hash\_joins - Post threshold hash joins” on page 1143

“post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144

“prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151

“priv\_workspace\_num\_overflows - Private Workspace Overflows” on page 1156

“priv\_workspace\_section\_inserts - Private Workspace Section Inserts” on page 1157

“priv\_workspace\_section\_lookups - Private Workspace Section Lookups” on page 1158

“priv\_workspace\_size\_top - Maximum Private Workspace Size” on page 1158

“rollback\_sql\_stmts - Rollback Statements Attempted” on page 1184

“rows\_deleted - Rows deleted monitor element” on page 1188

“rows\_inserted - Rows inserted monitor element” on page 1189

“rows\_read - Rows read monitor element” on page 1191

“rows\_selected - Rows Selected” on page 1195

“rows\_updated - Rows updated monitor element” on page 1196

“sec\_log\_used\_top - Maximum Secondary Log Space Used” on page 1199

“select\_sql\_stmts - Select SQL Statements Executed” on page 1203

“server\_platform - Server Operating System” on page 1206

“shr\_workspace\_num\_overflows - Shared Workspace Overflows” on page 1211

“shr\_workspace\_section\_inserts - Shared Workspace Section Inserts” on page 1212

“shr\_workspace\_section\_lookups - Shared Workspace Section Lookups” on page 1213

“shr\_workspace\_size\_top - Maximum Shared Workspace Size” on page 1213

“sort\_overflows - Sort overflows monitor element” on page 1224

“static\_sql\_stmts - Static SQL Statements Attempted” on page 1241

“stats\_cache\_size - Size of statistics cache monitor element” on page 1242

“stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element” on page 1243

“stats\_fabrications - Total number of statistics fabrications monitor elements” on page 1244

“sync\_runstats - Total number of synchronous RUNSTATS activities monitor element” on page 1265

“sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element” on page 1266

“tot\_log\_used\_top - Maximum Total Log Space Used” on page 1310

“total\_cons - Connects Since Database Activation” on page 1324

“total\_hash\_joins - Total Hash Joins” on page 1338

“total\_hash\_loops - Total Hash Loops” on page 1338

“total\_olap\_funcs - Total OLAP Functions monitor element” on page 1346

“total\_sort\_time - Total sort time monitor element” on page 1374

“total\_sorts - Total sorts monitor element” on page 1375

"uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed" on page 1398  
 "unread\_prefetch\_pages - Unread prefetch pages monitor element" on page 1399  
 "x\_lock\_escals - Exclusive lock escalations monitor element" on page 1429  
 "xquery\_stmts - XQuery Statements Attempted" on page 1431  
 "elapsed\_exec\_time - Statement Execution Elapsed Time" on page 806  
 "num\_log\_part\_page\_io - Number of Partial Log Page Writes" on page 977  
 "pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element" on page 1038  
 "pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element" on page 1038  
 "pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element" on page 1039  
 "pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123  
 "pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125  
 "pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136  
 "pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139  
 "pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141  
 "sort\_shrheap\_top - Sort share heap high watermark" on page 1226  
 CAT\_CACHE\_HEAP\_FULL  
 LOG\_FILE\_ARCHIVE  
 LOG\_FILE\_NUM\_CURR  
 LOG\_FILE\_NUM\_FIRST  
 LOG\_FILE\_NUM\_LAST  
 NUM\_LOG\_BUFF\_FULL  
 NUM\_LOG\_DATA\_IN\_BUFF

#### **event\_dbheader logical data group**

"conn\_time - Time of database connection monitor element" on page 736  
 "db\_name - Database name monitor element" on page 773  
 "db\_path - Database Path" on page 774

#### **event\_dbmemuse logical data group**

"node\_number - Node Number" on page 970  
 "pool\_config\_size - Configured Size of Memory Pool" on page 1040  
 "pool\_cur\_size - Current Size of Memory Pool" on page 1041  
 "pool\_id - Memory Pool Identifier" on page 1071  
 "pool\_watermark - Memory Pool Watermark" on page 1127  
 "evmon\_activates - Number of event monitor activations" on page 813  
 "evmon\_flushes - Number of Event Monitor Flushes" on page 819  
 "pool\_secondary\_id - Memory Pool Secondary Identifier" on page 1114  
 POOL\_MAX\_SIZE

### **event\_deadlock\_logical\_data\_group**

- “deadlock\_id - Deadlock Event Identifier” on page 782
- “deadlock\_node - Partition Number Where Deadlock Occurred” on page 783
- “dl\_conns - Connections involved in deadlock monitor element” on page 804
- “evmon\_activates - Number of event monitor activations” on page 813
- “rolled\_back\_agent\_id - Rolled Back Agent” on page 1185
- “rolled\_back\_appl\_id - Rolled Back Application” on page 1186
- “rolled\_back\_participant\_no - Rolled back application participant monitor element” on page 1186
- “rolled\_back\_sequence\_no - Rolled Back Sequence Number” on page 1186
- “start\_time - Event Start Time” on page 1241

### **event\_detailed\_dlconn\_logical\_data\_group**

- “agent\_id - Application handle (agent ID) monitor element” on page 650
- “appl\_id - Application ID monitor element” on page 666
- “appl\_id\_holding\_lk - Application ID Holding Lock” on page 668
- “blocking\_cursor - Blocking Cursor” on page 691
- “consistency\_token - Package consistency token monitor element” on page 738
- “creator - Application Creator” on page 763
- “cursor\_name - Cursor Name” on page 766
- “data\_partition\_id - Data partition identifier monitor element” on page 767
- “deadlock\_id - Deadlock Event Identifier” on page 782
- “deadlock\_node - Partition Number Where Deadlock Occurred” on page 783
- “evmon\_activates - Number of event monitor activations” on page 813
- “lock\_escalation - Lock escalation monitor element” on page 904
- “lock\_mode - Lock mode monitor element” on page 912
- “lock\_mode\_requested - Lock mode requested monitor element” on page 913
- “lock\_node - Lock Node” on page 915
- “lock\_object\_name - Lock Object Name” on page 915
- “lock\_object\_type - Lock object type waited on monitor element” on page 916
- “lock\_wait\_start\_time - Lock wait start timestamp monitor element” on page 923
- “locks\_held - Locks held monitor element” on page 932
- “locks\_in\_list - Number of Locks Reported” on page 933
- “package\_name - Package name monitor element” on page 1006
- “package\_version\_id - Package version monitor element” on page 1007
- “participant\_no - Participant within Deadlock” on page 1015
- “participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application” on page 1015
- “section\_number - Section number monitor element” on page 1201
- “sequence\_no - Sequence number monitor element” on page 1204
- “sequence\_no\_holding\_lk - Sequence Number Holding Lock” on page 1204
- “start\_time - Event Start Time” on page 1241
- “stmt\_operation/operation - Statement operation monitor element” on page 1250
- “stmt\_text - SQL statement text monitor element” on page 1255

"stmt\_type - Statement type monitor element" on page 1256  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"tablespace\_name - Table space name monitor element" on page 1279  
"lock\_attributes - Lock attributes monitor element" on page 901  
"lock\_count - Lock count monitor element" on page 902  
"lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
"lock\_hold\_count - Lock hold count monitor element" on page 911  
"lock\_name - Lock name monitor element" on page 914  
"lock\_release\_flags - Lock release flags monitor element" on page 918  
"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389

#### **event\_dlconn logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_id\_holding\_lk - Application ID Holding Lock" on page 668  
"data\_partition\_id - Data partition identifier monitor element" on page 767  
"deadlock\_id - Deadlock Event Identifier" on page 782  
"deadlock\_node - Partition Number Where Deadlock Occurred" on page 783  
"evmon\_activates - Number of event monitor activations" on page 813  
"lock\_attributes - Lock attributes monitor element" on page 901  
"lock\_count - Lock count monitor element" on page 902  
"lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
"lock\_escalation - Lock escalation monitor element" on page 904  
"lock\_hold\_count - Lock hold count monitor element" on page 911  
"lock\_mode - Lock mode monitor element" on page 912  
"lock\_mode\_requested - Lock mode requested monitor element" on page 913  
"lock\_name - Lock name monitor element" on page 914  
"lock\_node - Lock Node" on page 915  
"lock\_object\_name - Lock Object Name" on page 915  
"lock\_object\_type - Lock object type waited on monitor element" on page 916  
"lock\_release\_flags - Lock release flags monitor element" on page 918  
"lock\_wait\_start\_time - Lock wait start timestamp monitor element" on page 923  
"participant\_no - Participant within Deadlock" on page 1015  
"participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application" on page 1015  
"sequence\_no - Sequence number monitor element" on page 1204

"sequence\_no\_holding\_lk - Sequence Number Holding Lock" on page 1204  
"start\_time - Event Start Time" on page 1241  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"tablespace\_name - Table space name monitor element" on page 1279  
"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389

#### **event\_histogrambin logical data group**

"bin\_id - Histogram bin identifier monitor element" on page 690  
"bottom - Histogram bin bottom monitor element" on page 692  
"histogram\_type - Histogram type monitor element" on page 868  
"number\_in\_bin - Number in bin monitor element" on page 983  
"service\_class\_id - Service class ID monitor element" on page 1207  
"statistics\_timestamp - Statistics timestamp monitor element" on page 1241  
"top - Histogram bin top monitor element" on page 1310  
"work\_action\_set\_id - Work action set ID monitor element" on page 1424  
"work\_class\_id - Work class ID monitor element" on page 1425  
"workload\_id - Workload ID monitor element" on page 1426  
"mon\_interval\_id - Monitor interval identifier monitor element" on page 967  
"member - Database member monitor element" on page 957

#### **event\_log\_header logical data group**

"byte\_order - Byte Order of Event Data" on page 697  
"codepage\_id - ID of Code Page Used by Application" on page 718  
"event\_monitor\_name - Event Monitor Name" on page 811  
"num\_nodes\_in\_db2\_instance - Number of Nodes in Partition" on page 980  
"server\_instance\_name - Server Instance Name" on page 1205  
"server\_prdid - Server Product/Version ID" on page 1206  
"territory\_code - Database Territory Code" on page 1303  
"version - Version of Monitor Data" on page 1419

#### **event\_overflow logical data group**

"count - Number of Event Monitor Overflows" on page 752  
"first\_overflow\_time - Time of First Event Overflow" on page 849  
"last\_overflow\_time - Time of Last Event Overflow" on page 895  
"node\_number - Node Number" on page 970

#### **event\_qstats logical data group**

"last\_wlm\_reset - Time of last reset monitor element" on page 897

"queue\_assignments\_total - Queue assignments total monitor element" on page 1165  
"queue\_size\_top - Queue size top monitor element" on page 1165  
"queue\_time\_total - Queue time total monitor element" on page 1166  
"service\_subclass\_name - Service subclass name monitor element" on page 1208  
"service\_superclass\_name - Service superclass name monitor element" on page 1209  
"statistics\_timestamp - Statistics timestamp monitor element" on page 1241  
"threshold\_domain - Threshold domain monitor element" on page 1305  
"threshold\_name - Threshold name monitor element" on page 1306  
"threshold\_predicate - Threshold predicate monitor element" on page 1307  
"thresholdid - Threshold ID monitor element" on page 1308  
"work\_action\_set\_name - Work action set name monitor element" on page 1424  
"work\_class\_name - Work class name monitor element" on page 1425  
"mon\_interval\_id - Monitor interval identifier monitor element" on page 967  
"member - Database member monitor element" on page 957

#### **event\_scstats logical data group**

"act\_cpu\_time\_top - Activity CPU time top monitor element" on page 638  
"act\_remapped\_in - Activities remapped in monitor element" on page 641  
"act\_remapped\_out - Activities remapped out monitor element" on page 641  
"act\_rows\_read\_top - Activity rows read top monitor element" on page 641  
"act\_throughput - Activity throughput monitor element" on page 643  
"agg\_temp\_tablespace\_top - Aggregate temporary table space top monitor element" on page 660  
"concurrent\_act\_top - Concurrent activity top monitor element" on page 725  
"concurrent\_wlo\_top - Concurrent workload occurrences top monitor element" on page 727  
"concurrent\_connection\_top - Concurrent connection top monitor element" on page 726  
"coord\_act\_aborted\_total - Coordinator activities aborted total monitor element" on page 741  
"coord\_act\_completed\_total - Coordinator activities completed total monitor element" on page 742  
"coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element" on page 742  
"coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element" on page 743  
"coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element" on page 744  
"coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element" on page 745  
"coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element" on page 746  
"coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element" on page 747  
"coord\_act\_rejected\_total - Coordinator activities rejected total monitor element" on page 748

"cost\_estimate\_top - Cost estimate top monitor element" on page 752  
 "cpu\_utilization - CPU utilization monitor element" on page 759  
 details\_xml (This XML document is a metrics document of type system\_metrics, as described in the XML schema document sqllib/misc/DB2MonCommon.xsd. Metrics reported for the default subclass SYSDEFAULTSUBCLASS under the superclass SYSDEFAULTSYSTEMCLASS have a value of 0.)  
 "last\_wlm\_reset - Time of last reset monitor element" on page 897  
 "request\_exec\_time\_avg - Request execution time average monitor element" on page 1183  
 "rows\_returned\_top - Actual rows returned top monitor element" on page 1194  
 "service\_class\_id - Service class ID monitor element" on page 1207  
 "service\_subclass\_name - Service subclass name monitor element" on page 1208  
 "service\_superclass\_name - Service superclass name monitor element" on page 1209  
 "statistics\_timestamp - Statistics timestamp monitor element" on page 1241  
 "temp\_tablespace\_top - Temporary table space top monitor element" on page 1303  
 "total\_cpu\_time - Total CPU time monitor element" on page 1330  
 "total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element" on page 1332  
 "uow\_completed\_total - Total completed units of work monitor element" on page 1400  
 "uow\_lifetime\_avg - Unit of work lifetime average monitor element" on page 1402  
 "uow\_throughput - Unit of work throughput monitor element" on page 1406  
 "uow\_total\_time\_top - UOW total time top monitor element" on page 1406  
 "app\_act\_aborted\_total - Total failed external coordinator activities monitor element" on page 661  
 "app\_act\_completed\_total - Total successful external coordinator activities monitor element" on page 662  
 "app\_act\_rejected\_total - Total rejected external coordinator activities monitor element" on page 664  
 "mon\_interval\_id - Monitor interval identifier monitor element" on page 967  
 "member - Database member monitor element" on page 957

#### **event\_start logical data group**

"start\_time - Event Start Time" on page 1241

#### **event\_stmt logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650  
 "agents\_top - Number of Agents Created" on page 658  
 "appl\_id - Application ID monitor element" on page 666  
 "blocking\_cursor - Blocking Cursor" on page 691  
 "consistency\_token - Package consistency token monitor element" on page 738  
 "creator - Application Creator" on page 763  
 "cursor\_name - Cursor Name" on page 766  
 "fetch\_count - Number of Successful Fetches" on page 847  
 "int\_rows\_deleted - Internal Rows Deleted" on page 884

*"int\_rows\_inserted - Internal Rows Inserted"* on page 884  
*"int\_rows\_updated - Internal Rows Updated"* on page 885  
*"package\_name - Package name monitor element"* on page 1006  
*"package\_version\_id - Package version monitor element"* on page 1007  
*"partial\_record - Partial Record monitor element"* on page 1014  
*"pool\_data\_l\_reads - Buffer pool data logical reads monitor element"* on page 1048  
*"pool\_data\_p\_reads - Buffer pool data physical reads monitor element"* on page 1050  
*"pool\_index\_l\_reads - Buffer pool index logical reads monitor element"* on page 1080  
*"pool\_index\_p\_reads - Buffer pool index physical reads monitor element"* on page 1082  
*"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element"* on page 1116  
*"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element"* on page 1118  
*"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element"* on page 1120  
*"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element"* on page 1121  
*"rows\_read - Rows read monitor element"* on page 1191  
*"rows\_written - Rows Written"* on page 1196  
*"section\_number - Section number monitor element"* on page 1201  
*"sequence\_no - Sequence number monitor element"* on page 1204  
*"sort\_overflows - Sort overflows monitor element"* on page 1224  
*"sql\_req\_id - Request Identifier for SQL Statement"* on page 1228  
*"sqlca - SQL Communications Area (SQLCA)"* on page 1229  
*"start\_time - Event Start Time"* on page 1241  
*"stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element"* on page 1243  
*"stmt\_operation/operation - Statement operation monitor element"* on page 1250  
*"stmt\_text - SQL statement text monitor element"* on page 1255  
*"stmt\_type - Statement type monitor element"* on page 1256  
*"stop\_time - Event Stop Time"* on page 1262  
*"sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element"* on page 1266  
*"system\_cpu\_time - System CPU time monitor element"* on page 1267  
*"total\_sort\_time - Total sort time monitor element"* on page 1374  
*"total\_sorts - Total sorts monitor element"* on page 1375  
*"user\_cpu\_time - User CPU time monitor element"* on page 1411  
*"evmon\_flushes - Number of Event Monitor Flushes"* on page 819  
*"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element"* on page 1123  
*"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element"* on page 1125

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

#### **event\_stmt\_history logical data group**

“comp\_env\_desc - Compilation environment monitor element” on page 722

“creator - Application Creator” on page 763

“deadlock\_id - Deadlock Event Identifier” on page 782

“deadlock\_node - Partition Number Where Deadlock Occurred” on page 783

“evmon\_activates - Number of event monitor activations” on page 813

“package\_name - Package name monitor element” on page 1006

“package\_version\_id - Package version monitor element” on page 1007

“participant\_no - Participant within Deadlock” on page 1015

“section\_number - Section number monitor element” on page 1201

“sequence\_no - Sequence number monitor element” on page 1204

“stmt\_first\_use\_time - Statement first use timestamp monitor element” on page 1246

“stmt\_history\_id - Statement history identifier” on page 1246

“stmt\_invocation\_id - Statement invocation identifier monitor element” on page 1247

“stmt\_isolation - Statement isolation” on page 1248

“stmt\_last\_use\_time - Statement last use timestamp monitor element” on page 1248

“stmt\_lock\_timeout - Statement lock timeout monitor element” on page 1249

“stmt\_nest\_level - Statement nesting level monitor element” on page 1249

“stmt\_pkgcache\_id - Statement package cache identifier monitor element” on page 1251

“stmt\_query\_id - Statement query identifier monitor element” on page 1252

“stmt\_source\_id - Statement source identifier” on page 1253

“stmt\_text - SQL statement text monitor element” on page 1255

“stmt\_type - Statement type monitor element” on page 1256

“appl\_id - Application ID monitor element” on page 666

#### **event\_subsection logical data group**

“agent\_id - Application handle (agent ID) monitor element” on page 650

“num\_agents - Number of Agents Working on a Statement” on page 971

“partial\_record - Partial Record monitor element” on page 1014

“ss\_exec\_time - Subsection Execution Elapsed Time” on page 1237

“ss\_node\_number - Subsection Node Number” on page 1238

“ss\_number - Subsection number monitor element” on page 1238

“ss\_sys\_cpu\_time - System CPU Time used by Subsection” on page 1239

“ss\_usr\_cpu\_time - User CPU Time used by Subsection” on page 1239

“tq\_max\_send\_spills - Maximum number of table queue buffers overflows” on page 1390

“tq\_rows\_read - Number of Rows Read from table queues” on page 1391

“tq\_rows\_written - Number of rows written to table queues” on page 1391

"*tq\_tot\_send\_spills* - Total number of table queue buffers overflowed monitor element" on page 1396

"*appl\_id* - Application ID monitor element" on page 666

"*evmon\_flushes* - Number of Event Monitor Flushes" on page 819

"*sql\_req\_id* - Request Identifier for SQL Statement" on page 1228

#### **event\_table logical data group**

"*data\_object\_pages* - Data Object Pages" on page 766

"*data\_partition\_id* - Data partition identifier monitor element" on page 767

"*event\_time* - Event Time" on page 811

"*evmon\_activates* - Number of event monitor activations" on page 813

"*evmon\_flushes* - Number of Event Monitor Flushes" on page 819

"*index\_object\_pages* - Index Object Pages" on page 875

"*lob\_object\_pages* - LOB Object Pages" on page 898

"*long\_object\_pages* - Long Object Pages" on page 941

"*overflow\_accesses* - Accesses to overflowed records monitor element" on page 1004

"*page\_reorgs* - Page reorganizations monitor element" on page 1009

"*partial\_record* - Partial Record monitor element" on page 1014

"*rows\_read* - Rows read monitor element" on page 1191

"*rows\_written* - Rows Written" on page 1196

"*table\_name* - Table name monitor element" on page 1269

"*table\_schema* - Table schema name monitor element" on page 1271

"*table\_type* - Table type monitor element" on page 1273

"*tablespace\_id* - Table space identification monitor element" on page 1276

"*xda\_object\_pages* - XDA Object Pages" on page 1430

#### **event\_tablespace logical data group**

"*direct\_read\_reqs* - Direct read requests monitor element" on page 790

"*direct\_read\_time* - Direct read time monitor element" on page 792

"*direct\_reads* - Direct reads from database monitor element" on page 794

"*direct\_write\_reqs* - Direct write requests monitor element" on page 796

"*direct\_write\_time* - Direct write time monitor element" on page 798

"*direct\_writes* - Direct writes to database monitor element" on page 800

"*event\_time* - Event Time" on page 811

"*evmon\_activates* - Number of event monitor activations" on page 813

"*evmon\_flushes* - Number of Event Monitor Flushes" on page 819

"*files\_closed* - Database files closed monitor element" on page 848

"*partial\_record* - Partial Record monitor element" on page 1014

"*pool\_async\_data\_read\_reqs* - Buffer pool asynchronous read requests monitor element" on page 1026

"*pool\_async\_data\_reads* - Buffer pool asynchronous data reads monitor element" on page 1027

"*pool\_async\_data\_writes* - Buffer pool asynchronous data writes monitor element" on page 1028

"*pool\_async\_index\_read\_reqs* - Buffer pool asynchronous index read requests monitor element" on page 1031

"pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element" on page 1032  
 "pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element" on page 1033  
 "pool\_async\_read\_time - Buffer Pool Asynchronous Read Time" on page 1033  
 "pool\_async\_write\_time - Buffer pool asynchronous write time monitor element" on page 1034  
 "pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048  
 "pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050  
 "pool\_data\_writes - Buffer pool data writes monitor element" on page 1052  
 "pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080  
 "pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082  
 "pool\_index\_writes - Buffer pool index writes monitor element" on page 1084  
 "pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element" on page 1087  
 "pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113  
 "pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116  
 "pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118  
 "pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120  
 "pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121  
 "pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128  
 "tablespace\_name - Table space name monitor element" on page 1279  
 "pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element" on page 1038  
 "pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element" on page 1038  
 "pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element" on page 1039  
 "pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136  
 "pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139  
 "pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141  
 TABLESPACE\_FS\_CACHING  
 "unread\_prefetch\_pages - Unread prefetch pages monitor element" on page 1399

### **event\_thresholdviolations logical data group**

- “activate\_timestamp - Activate timestamp monitor element” on page 644
- “activity\_collected - Activity collected monitor element” on page 645
- “activity\_id - Activity ID monitor element” on page 645
- “agent\_id - Application handle (agent ID) monitor element” on page 650
- “appl\_id - Application ID monitor element” on page 666
- “coord\_partition\_num - Coordinator partition number monitor element” on page 750
- “destination\_service\_class\_id - Destination service class ID monitor element” on page 787
- “source\_service\_class\_id - Source service class ID monitor element” on page 1227
- “threshold\_action - Threshold action monitor element” on page 1305
- “threshold\_maxvalue - Threshold maximum value monitor element” on page 1306
- “threshold\_predicate - Threshold predicate monitor element” on page 1307
- “threshold\_queuesize - Threshold queue size monitor element” on page 1308
- “thresholdid - Threshold ID monitor element” on page 1308
- “time\_of\_violation - Time of violation monitor element” on page 1309
- “uow\_id - Unit of work ID monitor element” on page 1401
- “member - Database member monitor element” on page 957

### **event\_wcstats logical data group**

- “act\_cpu\_time\_top - Activity CPU time top monitor element” on page 638
- “act\_rows\_read\_top - Activity rows read top monitor element” on page 641
- “act\_total - Activities total monitor element” on page 643
- “coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element” on page 742
- “coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element” on page 743
- “coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element” on page 744
- “coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element” on page 745
- “coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element” on page 746
- “coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element” on page 747
- “cost\_estimate\_top - Cost estimate top monitor element” on page 752
- “last\_wlm\_reset - Time of last reset monitor element” on page 897
- “rows\_returned\_top - Actual rows returned top monitor element” on page 1194
- “statistics\_timestamp - Statistics timestamp monitor element” on page 1241
- “temp\_tablespace\_top - Temporary table space top monitor element” on page 1303
- “work\_action\_set\_id - Work action set ID monitor element” on page 1424
- “work\_action\_set\_name - Work action set name monitor element” on page 1424
- “work\_class\_id - Work class ID monitor element” on page 1425

“work\_class\_name - Work class name monitor element” on page 1425  
“mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
“member - Database member monitor element” on page 957

#### **event\_wlstats logical data group**

“act\_cpu\_time\_top - Activity CPU time top monitor element” on page 638  
“act\_rows\_read\_top - Activity rows read top monitor element” on page 641  
“act\_throughput - Activity throughput monitor element” on page 643  
“concurrent\_wlo\_act\_top - Concurrent WLO activity top monitor element” on page 726  
“concurrent\_wlo\_top - Concurrent workload occurrences top monitor element” on page 727  
“coord\_act\_aborted\_total - Coordinator activities aborted total monitor element” on page 741  
“coord\_act\_completed\_total - Coordinator activities completed total monitor element” on page 742  
“coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element” on page 742  
“coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element” on page 743  
“coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element” on page 744  
“coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element” on page 745  
“coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element” on page 746  
“coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element” on page 747  
“coord\_act\_rejected\_total - Coordinator activities rejected total monitor element” on page 748  
“cost\_estimate\_top - Cost estimate top monitor element” on page 752  
“cpu\_utilization - CPU utilization monitor element” on page 759  
details\_xml (This XML document is a metrics document of type system\_metrics, as described in the XML schema document sqllib/misc/DB2MonCommon.xsd.)  
“last\_wlm\_reset - Time of last reset monitor element” on page 897  
“lock\_wait\_time\_top - Lock wait time top monitor element” on page 928  
“rows\_returned\_top - Actual rows returned top monitor element” on page 1194  
“statistics\_timestamp - Statistics timestamp monitor element” on page 1241  
“temp\_tablespace\_top - Temporary table space top monitor element” on page 1303  
“total\_cpu\_time - Total CPU time monitor element” on page 1330  
“total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332  
“uow\_completed\_total - Total completed units of work monitor element” on page 1400  
“uow\_lifetime\_avg - Unit of work lifetime average monitor element” on page 1402  
“uow\_throughput - Unit of work throughput monitor element” on page 1406

"uow\_total\_time\_top - UOW total time top monitor element" on page 1406  
"wlo\_completed\_total - Workload occurrences completed total monitor element" on page 1424  
"workload\_id - Workload ID monitor element" on page 1426  
"workload\_name - Workload name monitor element" on page 1427  
"app\_act\_aborted\_total - Total failed external coordinator activities monitor element" on page 661  
"app\_act\_completed\_total - Total successful external coordinator activities monitor element" on page 662  
"app\_act\_rejected\_total - Total rejected external coordinator activities monitor element" on page 664  
"lock\_wait\_time\_global\_top - Top global lock wait time monitor element" on page 928  
"mon\_interval\_id - Monitor interval identifier monitor element" on page 967  
"member - Database member monitor element" on page 957

#### **event\_xact logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"lock\_escals - Number of lock escalations monitor element" on page 905  
"lock\_wait\_time - Time waited on locks monitor element" on page 924  
"locks\_held\_top - Maximum number of locks held monitor element" on page 932  
"partial\_record - Partial Record monitor element" on page 1014  
"prev\_uow\_stop\_time - Previous Unit of Work Completion Timestamp" on page 1155  
"rows\_read - Rows read monitor element" on page 1191  
"rows\_written - Rows Written" on page 1196  
"sequence\_no - Sequence number monitor element" on page 1204  
"system\_cpu\_time - System CPU time monitor element" on page 1267  
"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389  
"uow\_log\_space\_used - Unit of work log space used monitor element" on page 1403  
"uow\_start\_time - Unit of work start timestamp monitor element" on page 1404  
"uow\_status - Unit of Work Status" on page 1405  
"stop\_time - Event Stop Time" on page 1262  
"user\_cpu\_time - User CPU time monitor element" on page 1411  
"x\_lock\_escals - Exclusive lock escalations monitor element" on page 1429  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819

### **evmonstart logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812
- “db2start\_time - Start Database Manager Timestamp” on page 772
- “db\_conn\_time - Database activation timestamp monitor element” on page 772

### **lock logical data group**

- “partition\_key - Partitioning key monitor element” on page 1016
- “dl\_conns - Connections involved in deadlock monitor element” on page 804
- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “event\_type - Event Type monitor element” on page 812
- “partition\_number - Partition Number” on page 1017
- “rolled\_back\_participant\_no - Rolled back application participant monitor element” on page 1186
- “deadlock\_type - Deadlock type monitor element” on page 783

### **lock\_activity\_values logical data group**

- “partition\_key - Partitioning key monitor element” on page 1016
- “activity\_id - Activity ID monitor element” on page 645
- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “event\_type - Event Type monitor element” on page 812
- “participant\_no - Participant within Deadlock” on page 1015
- “partition\_number - Partition Number” on page 1017
- “stmt\_value\_index - Value index” on page 1259
- “stmt\_value\_isnull - Value has null value monitor element” on page 1260
- “stmt\_value\_isreopt - Variable used for statement reoptimization monitor element” on page 1261
- “stmt\_value\_type - Value type monitor element” on page 1261
- “uow\_id - Unit of work ID monitor element” on page 1401
- “stmt\_value\_data - Value data” on page 1259
- “event\_id - Event ID monitor element” on page 810

### **lock\_participant\_activities logical data group**

- “partition\_key - Partitioning key monitor element” on page 1016
- “activity\_id - Activity ID monitor element” on page 645
- “activity\_type - Activity type monitor element” on page 647
- “consistency\_token - Package consistency token monitor element” on page 738
- “effective\_isolation - Effective isolation monitor element” on page 805
- “effective\_query\_degree - Effective query degree monitor element” on page 806
- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “event\_type - Event Type monitor element” on page 812

“incremental\_bind - Incremental bind monitor element” on page 874  
 “package\_name - Package name monitor element” on page 1006  
 “package\_schema - Package schema monitor element” on page 1007  
 “package\_version\_id - Package version monitor element” on page 1007  
 “participant\_no - Participant within Deadlock” on page 1015  
 “partition\_number - Partition Number” on page 1017  
 “query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element” on page 1162  
 “reopt - Reopt bind option monitor element” on page 1176  
 “section\_number - Section number monitor element” on page 1201  
 “stmt\_first\_use\_time - Statement first use timestamp monitor element” on page 1246  
 “stmt\_invocation\_id - Statement invocation identifier monitor element” on page 1247  
 “stmt\_last\_use\_time - Statement last use timestamp monitor element” on page 1248  
 “stmt\_lock\_timeout - Statement lock timeout monitor element” on page 1249  
 “stmt\_nest\_level - Statement nesting level monitor element” on page 1249  
 “stmt\_pkgcache\_id - Statement package cache identifier monitor element” on page 1251  
 “stmt\_query\_id - Statement query identifier monitor element” on page 1252  
 “stmt\_source\_id - Statement source identifier” on page 1253  
 “stmt\_type - Statement type monitor element” on page 1256  
 “uow\_id - Unit of work ID monitor element” on page 1401  
 “stmt\_text - SQL statement text monitor element” on page 1255  
 “stmt\_unicode - Statement unicode flag monitor element” on page 1258  
 “stmt\_operation/operation - Statement operation monitor element” on page 1250

#### **lock\_participants logical data group**

“partition\_key - Partitioning key monitor element” on page 1016  
 “agent\_status - DCS Application Agents” on page 652  
 “agent\_id - Application handle (agent ID) monitor element” on page 650  
 “appl\_id - Application ID monitor element” on page 666  
 “appl\_name - Application name monitor element” on page 670  
 “application\_handle - Application handle monitor element” on page 675  
 “auth\_id - Authorization ID” on page 686  
 “client\_acctng - Client accounting string monitor element” on page 709  
 “client\_applname - Client application name monitor element” on page 710  
 “client\_userid - Client user ID monitor element” on page 717  
 “client\_wrkstname - Client workstation name monitor element” on page 717  
 “coord\_agent\_tid - Coordinator agent engine dispatchable unit ID monitor element” on page 747  
 “current\_request - Current operation request monitor element” on page 766  
 “event\_id - Event ID monitor element” on page 810  
 “event\_timestamp - Event timestamp monitor element” on page 811  
 “event\_type - Event Type monitor element” on page 812

"lock\_attributes - Lock attributes monitor element" on page 901  
 "lock\_count - Lock count monitor element" on page 902  
 "lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
 "lock\_escalation - Lock escalation monitor element" on page 904  
 "lock\_hold\_count - Lock hold count monitor element" on page 911  
 "lock\_mode - Lock mode monitor element" on page 912  
 "lock\_mode\_requested - Lock mode requested monitor element" on page 913  
 "lock\_name - Lock name monitor element" on page 914  
 "lock\_object\_type - Lock object type waited on monitor element" on page 916  
 LOCK\_OBJECT\_TYPE\_ID  
 "lock\_release\_flags - Lock release flags monitor element" on page 918  
 LOCK\_RRIID  
 "lock\_status - Lock status monitor element" on page 919  
 "lock\_timeout\_val - Lock timeout value monitor element" on page 920  
 "lock\_wait\_end\_time - Lock wait end timestamp monitor element" on page 923  
 "lock\_wait\_start\_time - Lock wait start timestamp monitor element" on page 923  
 "lock\_wait\_val - Lock wait value monitor element" on page 928  
 "member - Database member monitor element" on page 957  
 "object\_requested - Requested object monitor element" on page 992  
 "participant\_no - Participant within Deadlock" on page 1015  
 "participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application" on page 1015  
 "participant\_type - Participant type monitor element" on page 1016  
 "past\_activities\_wrapped - Past activities list wrapped monitor element" on page 1018  
 "service\_class\_id - Service class ID monitor element" on page 1207  
 "service\_subclass\_name - Service subclass name monitor element" on page 1208  
 "table\_file\_id - Table file ID monitor element" on page 1268  
 "table\_name - Table name monitor element" on page 1269  
 "table\_schema - Table schema name monitor element" on page 1271  
 "thresholdid - Threshold ID monitor element" on page 1308  
 "threshold\_name - Threshold name monitor element" on page 1306  
 "workload\_id - Workload ID monitor element" on page 1426  
 "workload\_name - Workload name monitor element" on page 1427  
 "agent\_tid - Agent thread ID monitor element" on page 653  
 "appl\_action - Application action monitor element" on page 665  
 "deadlock\_member - Deadlock member monitor element" on page 782  
 "queue\_start\_time - Queue start timestamp monitor element" on page 1165  
 "queued\_agents - Queued threshold agents monitor element" on page 1166  
 "service\_superclass\_name - Service superclass name monitor element" on page 1209  
 "tablespace\_name - Table space name monitor element" on page 1279  
 "xid - Transaction ID" on page 1431  
 "utility\_invocation\_id - Utility invocation ID" on page 1413

## **pkgcache logical data group**

- "partition\_key - Partitioning key monitor element" on page 1016
- "comp\_env\_desc - Compilation environment monitor element" on page 722
- "effective\_isolation - Effective isolation monitor element" on page 805
- "event\_id - Event ID monitor element" on page 810
- "event\_timestamp - Event timestamp monitor element" on page 811
- "executable\_id - Executable ID monitor element" on page 818
- "insert\_timestamp - Insert timestamp monitor element" on page 878
- "last\_metrics\_update - Metrics last update timestamp monitor element" on page 895
- "member - Database member monitor element" on page 957
- "num\_coord\_exec - Number of executions by coordinator agent monitor element" on page 972
- "num\_coord\_exec\_with\_metrics - Number of executions by coordinator agent with metrics monitor element" on page 972
- "num\_exec\_with\_metrics - Number of executions with metrics collected monitor element" on page 974
- "num\_executions - Statement executions monitor element" on page 973
- "package\_name - Package name monitor element" on page 1006
- "package\_schema - Package schema monitor element" on page 1007
- "package\_version\_id - Package version monitor element" on page 1007
- "partition\_number - Partition Number" on page 1017
- "prep\_time - Preparation time monitor element" on page 1154
- "query\_cost\_estimate - Query cost estimate monitor element" on page 1163
- "query\_data\_tag\_list - Estimated query data tag list monitor element" on page 1164
- "routine\_id - Routine ID monitor element" on page 1187
- "section\_env - Section environment monitor element" on page 1201
- "section\_number - Section number monitor element" on page 1201
- "section\_type - Section type indicator monitor element" on page 1202
- "stmt\_pkgcache\_id - Statement package cache identifier monitor element" on page 1251
- "stmt\_type\_id - Statement type identifier monitor element" on page 1257
- "total\_stats\_fabrication\_time - Total statistics fabrication time monitor element" on page 1378
- "total\_stats\_fabrications - Total statistics fabrications monitor elements" on page 1379
- "total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element" on page 1382
- "total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements" on page 1380
- "stmt\_text - SQL statement text monitor element" on page 1255
- "max\_coord\_stmt\_exec\_time - Maximum coordinator statement execution time monitor element" on page 943
- "max\_coord\_stmt\_exec\_timestamp - Maximum coordinator statement execution timestamp monitor element" on page 945

## **pkgcache\_metrics logical data group**

- “partition\_key - Partitioning key monitor element” on page 1016
- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “partition\_number - Partition Number” on page 1017
- “wlm\_queue\_time\_total - Workload manager total queue time monitor element” on page 1422
- “wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element” on page 1421
- “fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element” on page 840
- “fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element” on page 824
- “fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element” on page 844
- “fcm\_message\_send\_wait\_time - FCM message send wait time monitor element” on page 828
- “lock\_wait\_time - Time waited on locks monitor element” on page 924
- “lock\_waits - Lock waits monitor element” on page 928
- “direct\_read\_time - Direct read time monitor element” on page 792
- “direct\_read\_reqs - Direct read requests monitor element” on page 790
- “direct\_write\_time - Direct write time monitor element” on page 798
- “direct\_write\_reqs - Direct write requests monitor element” on page 796
- “log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933
- “num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element” on page 975
- “log\_disk\_wait\_time - Log disk wait time monitor element” on page 935
- “log\_disk\_waits\_total - Total log disk waits monitor element” on page 936
- “pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128
- “pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113
- “audit\_file\_write\_wait\_time - Audit file write wait time monitor element” on page 680
- “audit\_file\_writes\_total - Total audit files written monitor element” on page 682
- “audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683
- “audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element” on page 685
- “diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element” on page 788
- “diaglog\_writes\_total - Total diagnostic log file writes monitor element” on page 789
- “fcm\_send\_wait\_time - FCM send wait time monitor element” on page 836
- “fcm\_rcv\_wait\_time - FCM received wait time monitor element” on page 832
- “total\_act\_wait\_time - Total activity wait time monitor element” on page 1312
- “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368

"total\_section\_sorts - Total section sorts monitor element" on page 1371  
"total\_section\_sort\_time - Total section sort time monitor element" on page 1370  
"total\_act\_time - Total activity time monitor element" on page 1311  
"rows\_read - Rows read monitor element" on page 1191  
"rows\_modified - Rows modified monitor element" on page 1190  
"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048  
"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080  
"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116  
"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120  
"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136  
"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123  
"total\_cpu\_time - Total CPU time monitor element" on page 1330  
"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050  
"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118  
"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139  
"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125  
"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082  
"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121  
"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052  
"pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141  
"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084  
"direct\_reads - Direct reads from database monitor element" on page 794  
"direct\_writes - Direct writes to database monitor element" on page 800  
"rows\_returned - Rows returned monitor element" on page 1193  
"deadlocks - Deadlocks detected monitor element" on page 783  
"lock\_timeouts - Number of lock timeouts monitor element" on page 920  
"lock\_escals - Number of lock escalations monitor element" on page 905  
"fcm\_sends\_total - FCM sends total monitor element" on page 838  
"fcm\_recvs\_total - FCM receives total monitor element" on page 834  
"fcm\_send\_volume - FCM send volume monitor element" on page 835  
"fcm\_rcv\_volume - FCM received volume monitor element" on page 831  
"fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830  
"fcm\_message\_recvs\_total - Total FCM message receives monitor element" on page 826

"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827

"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823

"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846

"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842

"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843

"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839

"tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396

"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"sort\_overflows - Sort overflows monitor element" on page 1224

"audit\_events\_total - Total audit events monitor element" on page 679

"total\_sorts - Total sorts monitor element" on page 1375

"stmt\_exec\_time - Statement execution time monitor element" on page 1245

"coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element" on page 751

"total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element" on page 1356

"total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements" on page 1356

"total\_section\_proc\_time - Total section processing time monitor element" on page 1367

"total\_app\_section\_executions - Total application section executions monitor element" on page 1317

"total\_section\_time - Total section time monitor element" on page 1373

"total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element" on page 1359

"total\_routine\_user\_code\_time - Total routine user code time monitor element" on page 1360

"total\_routine\_time - Total routine time monitor element" on page 1357

"thresh\_violations - Number of threshold violations monitor element" on page 1304

"num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element" on page 979

"total\_routine\_invocations - Total routine invocations monitor elements" on page 1355

"lock\_wait\_time\_global - Lock wait time global monitor element" on page 926

"lock\_waits\_global - Lock waits global monitor element" on page 930

"reclaim\_wait\_time - Reclaim wait time monitor element" on page 1170

"spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element" on page 1236

"lock\_timeouts\_global - Lock timeouts global monitor element" on page 922

"lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element" on page 910

"lock\_escals\_locklist - Number of locklist lock escalations monitor element" on page 909

"lock\_escals\_global - Number of global lock escalations monitor element" on page 907

"cf\_wait\_time - cluster caching facility wait time monitor element" on page 703

"cf\_waits - Number of cluster caching facility waits monitor element" on page 703

"pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element" on page 1044

"pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element" on page 1045

"pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element" on page 1047

"pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element" on page 1042

"pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element" on page 1075

"pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements" on page 1077

"pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element" on page 1078

"pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element" on page 1074

"pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element" on page 1132

"pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element" on page 1134

"pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element" on page 1138

"pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element" on page 1131

"evmon\_wait\_time - Event monitor wait time monitor element" on page 814

"evmon\_waits\_total - Event monitor total waits monitor element" on page 816

"total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element" on page 1335

"total\_extended\_latch\_waits - Total extended latch waits monitor element" on page 1336

"total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element" on page 1332

"pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element" on page 1089

"pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element" on page 1093

"pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element" on page 1111

"pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element" on page 1087

“pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element” on page 1091

“pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element” on page 1109

### **pkgcache\_stmt\_args logical data group**

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“stmt\_value\_index - Value index” on page 1259

“stmt\_value\_isreopt - Variable used for statement reoptimization monitor element” on page 1261

“stmt\_value\_isnull - Value has null value monitor element” on page 1260

“stmt\_value\_type - Value type monitor element” on page 1261

“stmt\_value\_data - Value data” on page 1259

“member - Database member monitor element” on page 957

### **regvar logical data group**

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“member - Database member monitor element” on page 957

“event\_type - Event Type monitor element” on page 812

“regvar\_name - Registry variable name” on page 1172

“regvar\_value - Registry variable value” on page 1173

“regvar\_old\_value - Registry variable old value” on page 1173

“regvar\_level - Registry variable level” on page 1172

“regvar\_collection\_type - Registry variable collection type” on page 1172

### **sqlca logical data group**

sqlcabc

sqlcaid

sqlcode

sqlerrd

sqlerrmc

sqlerrml

sqlerrp

sqlstate

sqlwarn

### **txncompletion logical data group**

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“member - Database member monitor element” on page 957

“event\_type - Event Type monitor element” on page 812

“global\_transaction\_id - Global transaction identifier monitor element” on page 851

“local\_transaction\_id - Local transaction identifier monitor element” on page 900

“savepoint\_id - Savepoint ID” on page 1198  
“uow\_id - Unit of work ID monitor element” on page 1401  
“ddl\_classification - DDL classification” on page 780  
“txn\_completion\_status - Transaction completion status” on page 1398

#### **uow logical data group**

“partition\_key - Partitioning key monitor element” on page 1016  
“application\_handle - Application handle monitor element” on page 675  
“appl\_id - Application ID monitor element” on page 666  
“appl\_name - Application name monitor element” on page 670  
“client\_acctng - Client accounting string monitor element” on page 709  
“client\_applname - Client application name monitor element” on page 710  
“client\_hostname - Client hostname monitor element” on page 711  
“client\_pid - Client process ID monitor element” on page 713  
“client\_port\_number - Client port number monitor element” on page 715  
“client\_prdid - Client product and version ID monitor element” on page 715  
“client\_userid - Client user ID monitor element” on page 717  
“client\_wrkstname - Client workstation name monitor element” on page 717  
“completion\_status - Completion status monitor element” on page 722  
“conn\_time - Time of database connection monitor element” on page 736  
“coord\_member - Coordinator member monitor element” on page 749  
“event\_id - Event ID monitor element” on page 810  
“event\_timestamp - Event timestamp monitor element” on page 811  
“executable\_list\_size - Size of executable list monitor element” on page 819  
“global\_transaction\_id - Global transaction identifier monitor element” on page 851  
“intra\_parallel\_state - Current state of intrapartition parallelism monitor element” on page 886  
“local\_transaction\_id - Local transaction identifier monitor element” on page 900  
“member - Database member monitor element” on page 957  
“db\_conn\_time - Database activation timestamp monitor element” on page 772  
“mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
“package\_list\_exceeded - Package list exceeded monitor element” on page 1006  
“package\_list\_size - Size of package list monitor element” on page 1006  
service\_class\_id - Service class ID  
service\_subclass\_name - Service subclass name  
service\_superclass\_name - Service superclass name  
“session\_auth\_id - Session authorization ID monitor element” on page 1211  
start\_time - Event start time  
stop\_time - Event stop time  
“system\_auth\_id - System authorization identifier monitor element” on page 1267  
UOW\_CLIENT\_PLATFORM  
UOW\_CLIENT\_PROTOCOL  
uow\_id - Unit of work ID

"uow\_log\_space\_used - Unit of work log space used monitor element" on page 1403

workload\_id - Workload ID

workload\_name - Workload name

workload\_occurrence\_id - Workload occurrence identifier

"client\_platform - Client operating platform monitor element" on page 714

"client\_protocol - Client communication protocol monitor element" on page 716

"executable\_list\_truncated - Executable list truncated monitor element" on page 819

"connection\_start\_time - Connection start time monitor element" on page 736

#### **uow\_executable\_list logical data group**

"partition\_key - Partitioning key monitor element" on page 1016

"appl\_id - Application ID monitor element" on page 666

"executable\_id - Executable ID monitor element" on page 818

"lock\_wait\_time - Time waited on locks monitor element" on page 924

"lock\_waits - Lock waits monitor element" on page 928

"num\_executions - Statement executions monitor element" on page 973

"partition\_number - Partition Number" on page 1017

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150

"rows\_read - Rows read monitor element" on page 1191

"sort\_overflows - Sort overflows monitor element" on page 1224

"total\_act\_time - Total activity time monitor element" on page 1311

"total\_act\_wait\_time - Total activity wait time monitor element" on page 1312

"total\_cpu\_time - Total CPU time monitor element" on page 1330

"total\_sorts - Total sorts monitor element" on page 1375

uow\_id - Unit of work ID

#### **uow\_metrics logical data group**

"partition\_key - Partitioning key monitor element" on page 1016

"appl\_id - Application ID monitor element" on page 666

"partition\_number - Partition Number" on page 1017

"uow\_id - Unit of work ID monitor element" on page 1401

"wlm\_queue\_time\_total - Workload manager total queue time monitor element" on page 1422

"wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element" on page 1421

"fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element" on page 840

"fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element" on page 824

"fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element" on page 844

"fcm\_message\_send\_wait\_time - FCM message send wait time monitor element" on page 828

"agent\_wait\_time - Agent wait time monitor element" on page 654  
 "agent\_waits\_total - Total agent waits monitor element" on page 655  
 "lock\_wait\_time - Time waited on locks monitor element" on page 924  
 "lock\_waits - Lock waits monitor element" on page 928  
 "direct\_read\_time - Direct read time monitor element" on page 792  
 "direct\_read\_reqs - Direct read requests monitor element" on page 790  
 "direct\_write\_time - Direct write time monitor element" on page 798  
 "direct\_write\_reqs - Direct write requests monitor element" on page 796  
 "log\_buffer\_wait\_time - Log buffer wait time monitor element" on page 933  
 "num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element" on page 975  
 "log\_disk\_wait\_time - Log disk wait time monitor element" on page 935  
 "log\_disk\_waits\_total - Total log disk waits monitor element" on page 936  
 "tcpip\_rcv\_wait\_time - TCP/IP received wait time monitor element" on page 1298  
 "tcpip\_rcvs\_total - TCP/IP receives total monitor element" on page 1299  
 "client\_idle\_wait\_time - Client idle wait time monitor element" on page 712  
 "ipc\_rcv\_wait\_time - Interprocess communication received wait time monitor element" on page 888  
 "ipc\_rcvs\_total - Interprocess communication receives total monitor element" on page 889  
 "ipc\_send\_wait\_time - Interprocess communication send wait time monitor element" on page 891  
 "ipc\_sends\_total - Interprocess communication send total monitor element" on page 892  
 "tcpip\_send\_wait\_time - TCP/IP send wait time monitor element" on page 1301  
 "tcpip\_sends\_total - TCP/IP sends total monitor element" on page 1302  
 "pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128  
 "pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113  
 "audit\_file\_write\_wait\_time - Audit file write wait time monitor element" on page 680  
 "audit\_file\_writes\_total - Total audit files written monitor element" on page 682  
 "audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element" on page 683  
 "audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element" on page 685  
 "diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element" on page 788  
 "diaglog\_writes\_total - Total diagnostic log file writes monitor element" on page 789  
 "fcm\_send\_wait\_time - FCM send wait time monitor element" on page 836  
 "fcm\_rcv\_wait\_time - FCM received wait time monitor element" on page 832  
 "total\_wait\_time - Total wait time monitor element" on page 1386  
 "rqsts\_completed\_total - Total requests completed monitor element" on page 1197  
 "total\_rqst\_time - Total request time monitor element" on page 1362

"app\_rqsts\_completed\_total - Total application requests completed monitor element" on page 665

"total\_app\_rqst\_time - Total application request time monitor element" on page 1316

"total\_section\_sort\_proc\_time - Total section sort processing time monitor element" on page 1368

"total\_section\_sorts - Total section sorts monitor element" on page 1371

"total\_section\_sort\_time - Total section sort time monitor element" on page 1370

"rows\_read - Rows read monitor element" on page 1191

"rows\_modified - Rows modified monitor element" on page 1190

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136

"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123

"total\_cpu\_time - Total CPU time monitor element" on page 1330

"act\_completed\_total - Total completed activities monitor element" on page 637

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139

"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_writes - Direct writes to database monitor element" on page 800

"rows\_returned - Rows returned monitor element" on page 1193

"deadlocks - Deadlocks detected monitor element" on page 783

"lock\_timeouts - Number of lock timeouts monitor element" on page 920

"lock\_escals - Number of lock escalations monitor element" on page 905

"fcm\_sends\_total - FCM sends total monitor element" on page 838

"fcm\_recvs\_total - FCM receives total monitor element" on page 834

"fcm\_send\_volume - FCM send volume monitor element" on page 835

"fcm\_rcv\_volume - FCM received volume monitor element" on page 831

"fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830

"fcm\_message\_rcvs\_total - Total FCM message receives monitor element" on page 826

"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827

"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823

"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846

"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842

"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843

"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839

"tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396

"tcpip\_send\_volume - TCP/IP send volume monitor element" on page 1300

"tcpip\_rcv\_volume - TCP/IP received volume monitor element" on page 1297

"ipc\_send\_volume - Interprocess communication send volume monitor element" on page 890

"ipc\_rcv\_volume - Interprocess communication received volume monitor element" on page 887

"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"sort\_overflows - Sort overflows monitor element" on page 1224

"audit\_events\_total - Total audit events monitor element" on page 679

"act\_rejected\_total - Total rejected activities monitor element" on page 640

"act\_aborted\_total - Total aborted activities monitor element" on page 636

"total\_sorts - Total sorts monitor element" on page 1375

"total\_routine\_time - Total routine time monitor element" on page 1357

"total\_compile\_proc\_time - Total compile processing time monitor element" on page 1322

"total\_compilations - Total compilations monitor element" on page 1321

"total\_compile\_time - Total compile time monitor element" on page 1323

"total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element" on page 1340

"total\_implicit\_compilations - Total implicit complications monitor element" on page 1339

"total\_implicit\_compile\_time - Total implicit compile time monitor element" on page 1341

"total\_runstats\_proc\_time - Total runtime statistics processing time monitor element" on page 1364

"total\_runstats - Total runtime statistics monitor element" on page 1363

“total\_runstats\_time - Total runtime statistics time monitor element” on page 1365

“total\_reorg\_proc\_time - Total reorganization processing time monitor element” on page 1350

“total\_reorgs - Total reorganizations monitor element” on page 1352

“total\_reorg\_time - Total reorganization time monitor element” on page 1351

“total\_load\_proc\_time - Total load processing time monitor element” on page 1342

“total\_loads - Total loads monitor element” on page 1344

“total\_load\_time - Total load time monitor element” on page 1343

“total\_section\_proc\_time - Total section processing time monitor element” on page 1367

“total\_app\_section\_executions - Total application section executions monitor element” on page 1317

“total\_section\_time - Total section time monitor element” on page 1373

“total\_commit\_proc\_time - Total commits processing time monitor element” on page 1319

“total\_app\_commits - Total application commits monitor elements” on page 1314

“total\_commit\_time - Total commit time monitor element” on page 1320

“total\_rollback\_proc\_time - Total rollback processing time monitor element” on page 1353

“total\_app\_rollbacks - Total application rollbacks monitor element” on page 1315

“total\_rollback\_time - Total rollback time monitor element” on page 1354

“total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element” on page 1359

“total\_routine\_user\_code\_time - Total routine user code time monitor element” on page 1360

“thresh\_violations - Number of threshold violations monitor element” on page 1304

“num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element” on page 979

“total\_routine\_invocations - Total routine invocations monitor elements” on page 1355

“int\_commits - Internal commits monitor element” on page 880

“int\_rollbacks - Internal rollbacks monitor element” on page 882

“cat\_cache\_inserts - Catalog cache inserts monitor element” on page 698

“cat\_cache\_lookups - Catalog cache lookups monitor element” on page 699

“pkg\_cache\_inserts - Package cache inserts monitor element” on page 1020

“pkg\_cache\_lookups - Package cache lookups monitor element” on page 1021

“act\_rqsts\_total - Total activity requests monitor elements” on page 642

“total\_act\_wait\_time - Total activity wait time monitor element” on page 1312

“total\_act\_time - Total activity time monitor element” on page 1311

“lock\_wait\_time\_global - Lock wait time global monitor element” on page 926

“lock\_waits\_global - Lock waits global monitor element” on page 930

“reclaim\_wait\_time - Reclaim wait time monitor element” on page 1170

“spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element” on page 1236

“lock\_timeouts\_global - Lock timeouts global monitor element” on page 922

“lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element” on page 910

“lock\_escals\_locklist - Number of locklist lock escalations monitor element” on page 909

“lock\_escals\_global - Number of global lock escalations monitor element” on page 907

“cf\_wait\_time - cluster caching facility wait time monitor element” on page 703

“cf\_waits - Number of cluster caching facility waits monitor element” on page 703

“pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element” on page 1044

“pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element” on page 1045

“pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element” on page 1047

“pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element” on page 1042

“pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element” on page 1075

“pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements” on page 1077

“pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element” on page 1078

“pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element” on page 1074

“pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element” on page 1132

“pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element” on page 1134

“pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element” on page 1138

“pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element” on page 1131

“evmon\_wait\_time - Event monitor wait time monitor element” on page 814

“evmon\_waits\_total - Event monitor total waits monitor element” on page 816

“total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335

“total\_extended\_latch\_waits - Total extended latch waits monitor element” on page 1336

“total\_stats\_fabrication\_proc\_time - Total statistics fabrication processing time monitor element” on page 1377

“total\_stats\_fabrications - Total statistics fabrications monitor elements” on page 1379

“total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378

“total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381

“total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element” on page 1382

“total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380

“total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332

“pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element” on page 1089

“pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element” on page 1093

“pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element” on page 1111

“pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element” on page 1087

“pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element” on page 1091

“pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element” on page 1109

“app\_act\_completed\_total - Total successful external coordinator activities monitor element” on page 662

“app\_act\_aborted\_total - Total failed external coordinator activities monitor element” on page 661

“app\_act\_rejected\_total - Total rejected external coordinator activities monitor element” on page 664

#### **uow\_package\_list logical data group**

“partition\_key - Partitioning key monitor element” on page 1016

“appl\_id - Application ID monitor element” on page 666

“invocation\_id - Invocation ID monitor element” on page 886

“nesting\_level - Nesting level monitor element” on page 967

“package\_elapsed\_time - Package elapsed time monitor element” on page 1005

“package\_id - Package identifier monitor element” on page 1005

“routine\_id - Routine ID monitor element” on page 1187

“uow\_id - Unit of work ID monitor element” on page 1401

“member - Database member monitor element” on page 957

#### **utillocation logical data group**

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“member - Database member monitor element” on page 957

“event\_type - Event Type monitor element” on page 812

“utility\_invocation\_id - Utility invocation ID” on page 1413

“utility\_type - Utility Type” on page 1417

“device\_type - Device type” on page 787

“location\_type - Location type” on page 901

“location - Location” on page 900

### **utilphase logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812
- “utility\_invocation\_id - Utility invocation ID” on page 1413
- “utility\_type - Utility Type” on page 1417
- “utility\_phase\_type - Utility phase type” on page 1415
- “phase\_start\_event\_id - Phase start event ID” on page 1018
- “phase\_start\_event\_timestamp - Phase start event timestamp” on page 1018
- “objtype - Object type monitor element” on page 996
- “object\_schema - Object schema monitor element” on page 992
- “object\_name - Object name monitor element” on page 991
- “utility\_phase\_detail - Utility phase detail” on page 1415

### **utilstart logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812
- “utility\_invocation\_id - Utility invocation ID” on page 1413
- “utility\_type - Utility Type” on page 1417
- “utility\_operation\_type - Utility operation type” on page 1414
- “utility\_invoker\_type - Utility Invoker Type” on page 1413
- “utility\_priority - Utility Priority” on page 1415
- “utility\_start\_type - Utility start type” on page 1416
- “objtype - Object type monitor element” on page 996
- “object\_schema - Object schema monitor element” on page 992
- “object\_name - Object name monitor element” on page 991
- “num\_tbsps - Number of table spaces monitor element” on page 981
- “tbsp\_names - Table space names” on page 1296
- “utility\_detail - Utility detail” on page 1412

### **utilstop logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812
- “utility\_invocation\_id - Utility invocation ID” on page 1413
- “utility\_type - Utility Type” on page 1417
- “utility\_stop\_type - Utility stop type” on page 1417
- “start\_event\_id - Start event ID” on page 1240
- “start\_event\_timestamp - Start event timestamp” on page 1240
- “sqlca - SQL Communications Area (SQLCA)” on page 1229

### **Target tables, control tables, and event monitor table management:**

You can define an event monitor so that it stores its event records in SQL tables. To do this, use the CREATE EVENT MONITOR statement with the WRITE TO TABLE clause.

When you create a write-to-table event monitor, the event monitor creates target tables to store records for each of the logical data groups returning data. In each table, the column names match the monitor element names that they represent. By default, the event monitor creates the tables in the event monitor creator's schema and names the tables by concatenating their corresponding logical data group name to the event monitor name.

For example, consider the following statement, which creates an event monitor that captures STATEMENTS events:

```
CREATE EVENT MONITOR test FOR STATEMENTS WRITE TO TABLE
```

Event monitors using the STATEMENTS event type collect data from the event\_connheader, event\_stmt, and event\_subsection logical data groups. Tables representing logical data groups that are specific to individual event types are created, along with a control table for every write-to-table event monitor. For the event monitor test, created by user riihi, the database manager creates the following tables:

- riihi.connheader\_test
- riihi.stmt\_test
- riihi.subsection\_test
- riihi.control\_test

The first three tables correspond to each of the logical data groups event\_connheader, event\_stmt, and event\_subsection. The last table, riihi.control\_test, is the control table. The control table contains event monitor metadata, specifically, from the event\_start, event\_dbheader (conn\_time monitor element only), and event\_overflow logical data groups.

Monitor elements are written to the overflow group only for *non-blocked* event monitors. With non-blocked event monitors, agents that generate events do not wait for the event buffers to be written to the table if the buffers are full. Instead, they discard monitor data coming from agents when data is coming faster than the event monitor can write the data. In this case, the event monitor records information in the control table to indicate that an overflow has taken place. Included in this information is the monitor element **message**, which in the event of an overflow contains the text OVERFLOW:n, where *n* represents the number of event records that were discarded because the event buffers were full.

Whenever a write-to-table event monitor is activated, it acquires an IN or IX table lock on each target table to prevent the table from being modified while the event monitor is active. Table locks are maintained on all tables while the event monitor is active. If exclusive access is required on any of the target tables (for example, to run a utility), deactivate the event monitor to release the table locks before attempting such access.

Each column name in a target table matches an event monitor element identifier. Any event monitor element that does not have a corresponding target table column is ignored.

You must manually prune write-to-table event monitor target tables, including the unformatted event (UE) tables. On highly active systems, event monitors can

quickly fill disk space because of the high volume of data that they record. Unlike defining event monitors that write to files or named pipes, you can define write-to-table event monitors to record information from only certain logical data groups or monitor elements. You can use this feature to collect only the data that is relevant to your purposes and reduce the volume of data that event monitors generate. For example, the following statement defines an event monitor that captures connection events only from the event\_conn logical data group and includes only the **lock\_waits** monitor element:

```
CREATE EVENT MONITOR conn_monitor FOR CONNECTIONS WRITE TO TABLE
CONN(INCLUDES(lock_waits))
```

You might not want to have the target tables for an event monitor in the default schema, with default table names, in the default table space. If you anticipate high volumes of monitoring data, you might want the target tables to exist in their own table space. You can specify the schema, table, and table space names for the CREATE EVENT MONITOR statement. The schema name and table name form a derived name for the table. You can add the table space name after the table name by using the optional IN clause. Unlike the target tables, which the DB2 database manager automatically creates, a table space must exist if you include it in an event monitor definition. If you do not specify a table space, a table space for which you have USE privileges is assigned.

A target table can be used by only a single event monitor. If you define a target table for another event monitor or if it cannot be created for any other reason, the CREATE EVENT MONITOR statement fails.

The table space name can be added after the table name with the optional IN clause. Unlike the target tables, which the DB2 database manager automatically creates, a table space must already exist if it is included in an event monitor definition. If no table space is specified, then a table space over which the definer has USE privileges will be assigned.

In a partitioned database environment, a write-to-table event monitor is active only on database partitions where the table space containing the event monitor table exists. If the target table space for an active event monitor does not exist on a particular database partition, the event monitor will be deactivated on that database partition, and an error is written to the **db2diag** command log file.

For increased performance in retrieving event monitor data, you can create indexes for the event tables. If you add table attributes such as triggers, relational integrity, and constraints, the event monitor ignores them.

For example, the following statement defines an event monitor that captures STATEMENTS events, using the event\_connheader, event\_stmt, and event\_subsection logical data groups. Each of the three target tables has different schema, table and table space combinations:

```
CREATE EVENT MONITOR test FOR STATEMENTS
WRITE TO TABLE CONNHEADER,
STMT (TABLE mydept.statements),
SUBSECTION (TABLE subsections, IN mytablespace)
```

Assuming that the user riihi issued the previous statement, the derived names and table spaces of the target tables are as follows:

- CONNHEADER: riihi.connheader\_test in the default table space
- STMT: mydept.statements in the default table space
- SUBSECTION: riihi.subsections in the mytablespace table space

If a target table does not exist when the event monitor activates, activation continues and data that would otherwise be inserted into the target table is ignored. Correspondingly, if a monitor element does not have a dedicated column in the target table, it is ignored.

For active write-to-table event monitors, there is a risk that the table spaces storing event records can reach their capacity. To control this risk for DMS table spaces, you can define the percentage of table space capacity at which the event monitor is deactivated. You can specify this value in the PCTDEACTIVATE clause for the CREATE EVENT MONITOR statement. For SMS table spaces, the value is set to 100. If you enabled the autoresize feature for the target table space, you should set the PCTDEACTIVATE value to 100.

In a non-partitioned database environment, all write-to-table event monitors are deactivated when the last application terminates (and the database has not been explicitly activated). In a partitioned database environment, write-to-table event monitors are deactivated when the catalog partition deactivates.

### Logical data groups and event monitor output tables:

Monitor elements that are frequently used together are grouped into *logical data groups*. Event monitors that write to tables generally produce one output table for each logical data group of monitor elements that they capture.

The following table presents the default target table names by event type.

Table 7. Write-to-table event monitor logical data groups

| Event type                          | Logical data group    | Information in logical group                          | Name of table to which elements belonging to logical group are written |
|-------------------------------------|-----------------------|---|--|
| DEADLOCKS <sup>1</sup>              | event_connheader      | Connection metadata.                                  | CONNHEADER_ <i>evmon-name</i>  |
|                                     | event_deadlock        | Deadlock data.  | DEADLOCK_ <i>evmon-name</i>  |
|                                     | event_dlconn          | Applications and locks that are involved in deadlock. | DLCONN_ <i>evmon-name</i>  |
|                                     | CONTROL <sup>2</sup>  | Event monitor metadata.                               | CONTROL_ <i>evmon-name</i>   |
| DEADLOCKS WITH DETAILS <sup>1</sup> | event_connheader      | Connection metadata.                                  | CONNHEADER_ <i>evmon-name</i>  |
|                                     | event_deadlock        | Deadlock data.  | DEADLOCK_ <i>evmon-name</i>  |
|                                     | event_detailed_dlconn | Applications that are involved in deadlock.           | DLCONN_ <i>evmon-name</i>  |
|                                     | dllock                | Locks that are involved in deadlock.                  | DLLOCK_ <i>evmon-name</i>  |
|                                     | CONTROL <sup>2</sup>  | Event monitor metadata.                               | CONTROL_ <i>evmon-name</i>   |

Table 7. Write-to-table event monitor logical data groups (continued)

| Event type   | Logical data group    | Information in logical group                         | Name of table to which elements belonging to logical group are written |
|--|-----------------------|--|--|
| DEADLOCKS WITH DETAILS HISTORY <sup>1</sup>        | event_connheader      | Connection metadata.                                 | CONNHEADER_ <i>evmon-name</i>  |
|  | event_deadlock        | Deadlock data.                                       | DEADLOCK_ <i>evmon-name</i>  |
|  | event_detailed_dlconn | Applications that are involved in deadlock.          | DLCONN_ <i>evmon-name</i>  |
|  | dllock                | Locks that are involved in deadlock.                 | DLLOCK_ <i>evmon-name</i>  |
|  | event_stmt            | List of the previous statements in the unit of work. | STMTHIST_ <i>evmon-name</i>  |
|  | CONTROL <sup>2</sup>  | Event monitor metadata.                              | CONTROL_ <i>evmon-name</i>   |
| DEADLOCKS WITH DETAILS HISTORY VALUES <sup>1</sup> | event_connheader      | Connection metadata.                                 | CONNHEADER_ <i>evmon-name</i>  |
|  | event_deadlock        | Deadlock data.                                       | DEADLOCK_ <i>evmon-name</i>  |
|  | event_detailed_dlconn | Applications that are involved in deadlock.          | DLCONN_ <i>evmon-name</i>  |
|  | dllock                | Locks that are involved in deadlock.                 | DLLOCK_ <i>evmon-name</i>  |
|  | event_stmt_history    | List of the previous statements in the unit of work. | STMTHIST_ <i>evmon-name</i>  |
|  | STMTVALS              | Input data values of statements in STMTHIST table.   | STMTVALS_ <i>evmon-name</i>  |
|  | CONTROL <sup>2</sup>  | Event monitor metadata.                              | CONTROL_ <i>evmon-name</i>   |
| STATEMENT  | event_connheader      | Connection metadata.                                 | CONNHEADER_ <i>evmon-name</i>  |
|  | event_stmt            | Statement data.                                      | STMT_ <i>evmon-name</i>  |
|  | event_subsection      | Statement data that is specific to subsection.       | SUBSECTION_ <i>evmon-name</i>  |
|  | CONTROL <sup>2</sup>  | Event monitor metadata.                              | CONTROL_ <i>evmon-name</i>   |
| TRANSACTIONS <sup>3</sup>                          | event_connheader      | Connection metadata.                                 | CONNHEADER_ <i>evmon-name</i>  |
|  | event_xact            | Transaction data.                                    | XACT_ <i>evmon-name</i>  |
|  | CONTROL <sup>2</sup>  | Event monitor metadata.                              | CONTROL_ <i>evmon-name</i>   |
| CONNECTIONS  | event_connheader      | Connection metadata.                                 | CONNHEADER_ <i>evmon-name</i>  |
|  | event_conn            | Connection data.                                     | CONN_ <i>evmon-name</i>  |
|  | CONTROL <sup>2</sup>  | Event monitor metadata.                              | CONTROL_ <i>evmon-name</i>   |
|  | event_connmemuse      | Memory pool metadata.                                | CONNMEMUSE_ <i>evmon-name</i>  |

Table 7. Write-to-table event monitor logical data groups (continued)

| Event type           | Logical data group        | Information in logical group   | Name of table to which elements belonging to logical group are written |
|----------------------|---------------------------|--|--|
| DATABASE             | event_db                  | Database manager data.   | DB_evmon-name  |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
|                      | event_dbmemuse            | Memory pool metadata.  | DBMEMUSE_evmon-name  |
| BUFFERPOOLS          | event_bufferpool          | Buffer pool data.  | BUFFERPOOL_evmon-name  |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
| TABLESPACES          | event_tablespace          | Table space data.  | TABLESPACE_evmon-name  |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
| TABLES               | event_table               | Table data.  | TABLE_evmon-name   |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
| ACTIVITIES           | event_activity            | Activities that completed executing or were captured in progress.  | ACTIVITY_evmon-name  |
|                      | event_activitystmt        | Statement information for activities that are statements.  | ACTIVITYSTMT_evmon-name  |
|                      | event_activityvals        | Input data values for activities that have them. The following data types are not reported: CLOB, REF, BOOLEAN, STRUCT, DATALINK, LONG VARGRAPHIC, LONG, XMLLOB, and DBCLOB. | ACTIVITYVALS_evmon-name  |
|                      | activity_metrics          | Activities metrics.  | ACTIVITYMETRICS_evmon-name   |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
| STATISTICS           | event_scstats             | Statistics that are computed from the activities that executed within each service class, work class, or workload in the system.   | SCSTATS_evmon-name   |
|                      | event_wcstats             |  | WCSTATS_evmon-name   |
|                      | event_wlstats             |  | WLSTATS_evmon-name   |
|                      | event_histogrambin        |  | HISTOGRAMBIN_evmon-name  |
|                      | event_qstats              |  | QSTATS_evmon-name  |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |
| THRESHOLD VIOLATIONS | event_thresholdviolations | List of thresholds that were violated and the times of violations.   | THRESHOLDVIOLATIONS_evmon-name   |
|                      | CONTROL <sup>2</sup>      | Event monitor metadata.  | CONTROL_evmon-name   |

Table 7. Write-to-table event monitor logical data groups (continued)

| Event type    | Logical data group          | Information in logical group   | Name of table to which elements belonging to logical group are written |
|---------------|-----------------------------|--|--|
| LOCKING       | lock                        | Summary lock wait, lock timeout or deadlock event information.   | LOCK_EVENT $_{evmon-name}$   |
|               | lock_participants           | Information about lock participants.   | LOCK_PARTICIPANTS_ $_{evmon-name}$                                     |
|               | lock_participant_activities | Activity data for each lock participant.   | LOCK_PARTICIPANT_ACTIVITIES_ $_{evmon-name}$                           |
|               | lock_activity_values        | Details about the specific data being processed by a specific activity.  | LOCK_ACTIVITY_VALUES_ $_{evmon-name}$                                  |
|               | CONTROL <sup>2</sup>        | Event monitor metadata.  | CONTROL_ $_{evmon-name}$   |
| PACKAGE CACHE | pkgcache                    | Summary package cache event information. This information includes detailed metrics in XML format in the METRICS column. | PKGCACHE_EVENT $_{evmon-name}$   |
|               | pkgcache_metrics            | Table containing the same metrics that are included in the METRICS column of the PKGCACHE table.                         | PKGCACHE_METRICS_ $_{evmon-name}$                                      |
|               | CONTROL <sup>2</sup>        | Event monitor metadata.  | CONTROL_ $_{evmon-name}$   |
| UNIT OF WORK  | uow                         | Summary unit of work event information. This information includes detailed metrics in XML format in the METRICS column.  | UOW_EVENT $_{evmon-name}$  |
|               | uow_metrics                 | Table containing the same metrics that are included in the METRICS column of the PKGCACHE table.                         | UOW_METRICS_ $_{evmon-name}$   |
|               | uow_package_list            | Package list detail information. <sup>4</sup>  | UOW_PACKAGE_LIST_ $_{evmon-name}$                                      |
|               | uow_executable_list         | Executable list information. <sup>4</sup>  | UOW_EXECUTABLE_LIST_ $_{evmon-name}$                                   |
|               | CONTROL <sup>2</sup>        | Event monitor metadata.  | CONTROL_ $_{evmon-name}$   |

Table 7. Write-to-table event monitor logical data groups (continued)

| Event type | Logical data group | Information in logical group  | Name of table to which elements belonging to logical group are written |
|------------|--------------------|---|--|
| 1          |                    | This option has been deprecated and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.  |  |
| 2          |                    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups.   |  |
| 3          |                    | This option has been deprecated and might be removed in a future release. Use the CREATE EVENT MONITOR FOR UNIT OF WORK statement to monitor transaction events.  |  |
| 4          |                    | Unless you explicitly specify which output tables to create for the unit of work event monitor, this table is included by default. If you do not set the configuration parameter for collecting the related information ( <code>mon_uow_pkglist</code> or <code>mon_uow_execlist</code> ) to ON, the table is created, but it contains no data. |  |

The following logical data groups are not collected for write-to-table event monitors:

- log\_stream\_header
- log\_header
- dbheader (only the `conn_time` monitor element is collected)

The data type of each column in an event monitor table corresponds to the data type of the monitor element represented by the column. The following table contains a set of data type mappings that correspond the original system monitor data types of the monitor elements (found in `sqlmon.h` file) to the SQL data types of the table columns.

Table 8. System Monitor Data Type Mappings

| System monitor data type             | SQL data type                |
|--------------------------------------|------------------------------|
| SQLM_TYPE_STRING                     | CHAR[n], VARCHAR[n], CLOB[n] |
| SQLM_TYPE_U8BIT and SQLM_TYPE_8BIT   | SMALLINT, INTEGER, or BIGINT |
| SQLM_TYPE_U16BIT and SQLM_TYPE_16BIT | SMALLINT, INTEGER, or BIGINT |
| SQLM_TYPE_U32BIT and SQLM_TYPE_32BIT | INTEGER or BIGINT            |
| SQLM_TYPE_U64BIT and SQLM_TYPE_64BIT | BIGINT                       |
| SQLM_TIMESTAMP                       | TIMESTAMP                    |
| SQLM_TIME                            | BIGINT                       |
| SQLCA: SQLERRMC                      | VARCHAR[72]                  |
| SQLCA: SQLSTATE                      | CHAR[5]                      |
| SQLCA: SQLWARN                       | CHAR[11]                     |
| SQLCA: other fields                  | INTEGER or BIGINT            |
| SQLM_TYPE_HANDLE                     | BLOB[n]                      |

**Note:**

1. All columns are NOT NULL.
2. Because the performance of tables with CLOB columns is inferior to tables that have VARCHAR columns, consider using the TRUNC keyword when specifying the `stmt evmGroup` (or `dlconn evmGroup`, when using deadlocks with details).

3. `SQLM_TYPE_HANDLE` is used to represent the compilation environment handle object.

### **Creating event monitors that write to unformatted event (UE) tables**

If the performance of event monitor data collection is particularly important, you might choose to have your event monitor write its output to an unformatted event (UE) table. Most of the data written to a UE table is written as inline binary data, which allows for faster I/O as data is collected.

Another advantage of UE tables over regular tables for event monitors is that you generally do not have to be concerned at event monitor creation time with different options such as what buffer size to use, whether the event monitor is blocked or unblocked, or what types of data (logical groups) must be collected. However, because most of the data collected is in binary format, you must post-process the UE table to be able to examine the event data.

**Note:** Starting in IBM DB2 10.1 Version 10.1, the following features related to the UE tables are available:

- You can use the procedure `EVMON_UPGRADE_TABLES` to upgrade the UE tables produced by event monitors in earlier releases. This capability lets you more easily retain event monitor data as you upgrade your DB2 product.
- All event monitors that can write their output to UE tables can also write to regular tables.
- You can prune unneeded data from UE tables using the option `PRUNE_UE_TABLE` of the procedure `EVMON_FORMAT_UE_TO_TABLES`.

### **Before you begin**

Keep the following considerations in mind when creating an event monitor that writes to an unformatted event table:

- You need `SQLADM` or `DBADM` authority to create an event monitor that writes to a UE table.
- Use a table space for your unformatted event tables that is optimized for performance. When you create the table space, keep the following guidelines in mind:
  - Specify a page size (`PAGESIZE`) as large as possible, up to 32KB. A large page size ensures that the BLOB containing the event data can be written inline with the table row. If the page size is too small to allow the BLOB to be inlined, performance of the event monitor might be diminished. The database manager attempts to inline the `event_data` BLOB column in the unformatted event table, but this is not always possible. To check that the rows in the unformatted event table have been inlined, use the `ADMIN_IS_INLINED` function. If the rows have not been inlined, use the `ADMIN_EST_INLINE_LENGTH` functions to determine how much space the rows need.
  - Specify the `NO FILE CACHING SYSTEM` option.
- In a partitioned database environment, consider on which partitions the table space exists. If a table space for a target unformatted event table does not exist on some database partition, data for that target unformatted event table is ignored. This behavior allows users to choose a subset of database partitions for monitoring to be chosen, by creating a table space that exists only on certain database partitions.

## About this task

The following event monitor types support the use of UE tables:

- Unit of work
- Package Cache
- Locking

**Note:** Despite their name, unformatted event tables are still relational tables. The main difference between a UE table produced by, say, a locking event monitor and a regular table produced by a locking event monitor is that most of the data in a UE table is written in binary format in the `EVENT_DATA` column. See “Unformatted event table column definitions” on page 98 for more information about the structure of UE tables.

## Procedure

To create an event monitor that writes to a UE table:

- Formulate a `CREATE EVENT MONITOR` statement, using the `WRITE TO UNFORMATTED EVENT TABLE` clause. For example, to create a unit of work event monitor called `uowmon`, you might use a statement like the one that follows:

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
```

By default, the name of the UE table that the event monitor creates is the same as the name of the event monitor.

- To specify an alternative to the default table name, use the `TABLE` clause. For example, if you want to have the UE table called `myunitsofwork`, construct the statement as follows:

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    TABLE myunitsofwork
```

You can also specify the table space in which to store the UE table using the `IN tablespace-name` clause:

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    TABLE myunitsofwork
                    IN mytablespace
```

or

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    IN mytablespace
```

The first example places the UE table `myunitsofwork` in table space `mytablespace`; the second example places a UE table named `uowmon` (the default, as no table name is specified) in table space `mytablespace`.

- By default, any event monitor that writes to a UE table is created to activate automatically on database activation. You can override this behaviour using the `MANUALSTART` clause:

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    MANUALSTART
```

In the preceding example, the event monitor `uowmon` must always be activated manually, using the `SET EVENT MONITOR STATE` statement.

## What to do next

By default, event monitors that were introduced in Version 9.7 or later are created as `AUTOSTART` event monitors. They are activated automatically when the database is next activated, and on subsequent database activations thereafter. If you want to activate the event monitor immediately, before the next database activation, use the `SET EVENT MONITOR STATE` statement to manually start the event monitor. In addition for each of the locking, unit of work and package cache event monitors, you must also enable data collection.

### Unformatted event table column definitions:

An unformatted event table is created when you issue a `CREATE EVENT MONITOR` statement that includes the clause `WRITE TO UNFORMATTED EVENT TABLE`. The column definitions are useful when you want to extract data to analyze or prune a table of unneeded data.

The column definitions for the unformatted event table are useful when you want to extract data from an unformatted event table using one of the following routines:

- `EVMON_FORMAT_UE_TO_XML` - extracts data from an unformatted event table into an XML document.
- `EVMON_FORMAT_UE_TO_TABLES` - extracts data from an unformatted event table into a set of relational tables.

The call to these routines accepts a `SELECT` statement that specifies the rows that you want to extract. Use the unformatted event table column definitions to assist with composing your `SELECT` statement.

There is no automatic purging of the event data written to an unformatted event table. You must manually purge data from the table. The column definitions for the unformatted event table are useful when you want to purge a targeted set of records. Another option is to remove all the table rows using the `TRUNCATE TABLE` statement.

As part of the `CREATE EVENT MONITOR` statement, you can specify what to name the associated unformatted event table. If not specified, the name defaults to the same name as the event monitor. The `SYSCAT.EVENTTABLES` catalog view lists event monitors, their associated unformatted table, and other details.

The following table describes the columns in the unformatted event table. The key column is the `event_data` column. The other columns represent identifiers that you can use to locate events of interest. For further attributes of table columns, issue a `DESCRIBE` statement.

*Table 9. Unformatted event table column definitions*

| Column name            | Column data type     | Column description  |
|------------------------|----------------------|---|
| <code>appl_id</code>   | <code>VARCHAR</code> | <code>appl_id</code> - Application ID monitor element     |
| <code>appl_name</code> | <code>VARCHAR</code> | <code>appl_name</code> - Application name monitor element |

Table 9. Unformatted event table column definitions (continued)

| Column name             | Column data type | Column description   |
|-------------------------|------------------|--|
| event_correlation_id    | BIT DATA         | An optional event correlation ID. A NULL value indicates that the event correlation ID was not available.<br><br>The value is based on the event monitor type: <ul style="list-style-type: none"> <li>• LOCKING - Reserved for future use</li> <li>• UOW- Reserved for future use</li> </ul> |
| event_data              | BLOB             | The entire event record data for an event captured by the event monitor, stored in its original binary form.   |
| event_id                | INTEGER          | event_id - Event ID monitor element  |
| event_timestamp         | TIMESTAMP        | event_timestamp - Event timestamp monitor element  |
| event_type              | VARCHAR          | event_type - Event Type monitor element monitor element  |
| member                  | SMALLINT         | member - Database member monitor element   |
| partitioning_key        | INTEGER          | The partitioning key for the table, so that insert operations are performed locally on the database partition where the event monitor is running.  |
| record_seq_num          | INTEGER          | The sequence number of the record that is stored within the event_data column.   |
| record_type             | INTEGER          | The type of record that is stored within the event_data column.  |
| service_subclass_name   | VARCHAR          | service_subclass_name - Service subclass name monitor element  |
| service_superclass_name | VARCHAR          | service_superclass_name - Service superclass name monitor element  |
| workload_name           | VARCHAR          | workload_name - Workload name monitor element  |
| mon_interval_id         | BIGINT           | mon_interval_id - Monitor interval identifier monitor element  |

**Differences between regular and UE table output:**

Generally speaking, the event monitors that can write to both regular and unformatted event (UE) tables capture the same data. However, there are some minor differences to be aware of.

### Order of columns

The first difference pertains to the order of the columns of the tables. When the event monitor generates regular tables, compared to the output produced by running `EVMON_FORMAT_UE_TO_TABLES` against a UE table, columns are generally presented in alphabetical order, with two exceptions:

- If a `PARTITION_KEY` column is included in the output, it is the first column.
- For tables that report metrics, related columns are grouped together. For example, columns that report time spent in the system are grouped together.

### Columns returned

The other pertains to data types of columns. In most cases, the columns in write-to-table event monitors are the same as the columns produced by running `EVMON_FORMAT_UE_TO_TABLES` against a UE table. There are some differences however. These differences are summarized in Table 10.

Table 10.

| Logical data group | Columns returned in regular tables   | Columns returned from <code>EVMON_FORMAT_UE_TO_TABLES</code>              |
|--------------------|--|---|
| All groups         | <code>PARTITION_KEY</code> column included                                 |   |
| uow                | <code>TYPE</code> column included  | <code>TYPE</code> column excluded   |
| uow_package_list   | <code>ROUTINE_ID</code> data type is <code>BIGINT</code>                   | <code>ROUTINE_ID</code> data type is <code>INTEGER</code>                 |
| pkgcache           | <code>XMLID</code> column excluded   | <code>XMLID</code> column included  |
| lock               | <code>DL_CONNS</code> data type is <code>BIGINT</code>                     | <code>DL_CONNS</code> data type is <code>INTEGER</code>                   |
|                    | <code>ROLLED_BACK_PARTICIPANT_NO</code> data type is <code>SMALLINT</code> | <code>ROLLED_BACK_PARTICIPANT_NO</code> data type is <code>INTEGER</code> |
|                    | <code>XMLID</code> column excluded   | <code>XMLID</code> column included  |
| lock_participants  | <code>AGENT_STATUS</code> data type is <code>BIGINT</code>                 | <code>AGENT_STATUS</code> data type is <code>INTEGER</code>               |
|                    | <code>APPL_ID</code> data type is <code>VARCHAR(64)</code>                 | <code>APPL_ID</code> data type is <code>VARCHAR(128)</code>               |
|                    | <code>APPL_NAME</code> data type is <code>VARCHAR(255)</code>              | <code>APPL_NAME</code> data type is <code>VARCHAR(128)</code>             |
|                    | <code>CLIENT_ACCTNG</code> data type is <code>VARCHAR(200)</code>          | <code>CLIENT_ACCTNG</code> data type is <code>VARCHAR(255)</code>         |
|                    | <code>TABLESPACE_NAME</code> data type is <code>VARCHAR(18)</code>         | <code>TABLESPACE_NAME</code> data type is <code>VARCHAR(128)</code>       |
|                    | <code>XMLID</code> column excluded   | <code>XMLID</code> column included  |
|                    | <code>INTERNAL_DATA</code> column included                                 | Data contained in <code>INTERNAL_DATA</code> is not included.             |

Table 10. (continued)

| Logical data group   | Columns returned in regular tables   | Columns returned from EVMON_FORMAT_UE_TO TABLES |
|--|--|---|
| lock_participant_activities  | ACTIVITY_ID data type is BIGINT  | ACTIVITY_ID data type is INTEGER                |
|  | Includes EVENT_ID, EVENT_TYPE and EVENT_TIMESTAMP rather than XMLID <sup>1</sup> | XMLID column included                           |
|  | CONSISTENCY_TOKEN is CHAR(8)   | CONSISTENCY_TOKEN is VARCHAR(8)                 |
| lock_activity_values   | ACTIVITY_ID data type is BIGINT  | ACTIVITY_ID data type is INTEGER                |
|  | PARTICIPANT_NO data type is SMALLINT   | PARTICIPANT_NO data type is INTEGER             |
|  | Includes EVENT_ID, EVENT_TYPE and EVENT_TIMESTAMP instead of XMLID. <sup>1</sup> | XMLID column included                           |
| <p>1. The XMLID column represents a compound monitor element made up of the concatenation of the event_header, event_id, event_type, event_timestamp and partition monitor elements.</p> |  |   |

## Creating a file event monitor

When creating an event monitor you must determine where the information it collects is to be stored. File event monitors store event records in files. File event monitors and their options are defined by the CREATE EVENT MONITOR statement.

### Before you begin

You will need SQLADM or DBADM authority to create a file event monitor.

### About this task

A file event monitor streams event records to a series of 8-character numbered files with the extension "evt" (for example, 00000000.evt, 00000001.evt, and 00000002.evt). The data should be considered to be one logical file even though the data is broken up into smaller pieces (that is, the start of the data stream is the first byte in the file 00000000.evt; the end of the data stream is the last byte in the file nnnnnnnn.evt). An event monitor will never span a single event record across two files.

### Procedure

1. Indicate that event monitor data is to be collected in a file (or set of files), and provide a directory location where event files are to be stored.

```
CREATE EVENT MONITOR d1mon FOR eventtype
        WRITE TO FILE '/tmp/d1events'
```

d1mon is the name of the event monitor.

/tmp/d1events is the name of the directory path (on UNIX systems) where the event monitor is to write the event files.

2. Specify the types of events to be monitored. You can monitor one or more event types with a single event monitor.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents'
```

This event monitor will monitor for the CONNECTIONS and DEADLOCKS WITH DETAILS event types.

3. Specify the size of the file event monitor buffers (in 4K pages) by adjusting the BUFFERSIZE value:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
```

8 is the capacity in 4K pages of the two event file buffers.

The default size of each buffer is 4 pages (two 16K buffers are allocated). The minimum size is 1 page. The maximum size of the buffers is limited by the size of the monitor heap, because the buffers are allocated from that heap. For performance reasons, highly active event monitors should have larger buffers than relatively inactive event monitors.

4. Indicate if you need the event monitor to be blocked or non-blocked. For blocked event monitors, each agent that generates an event will wait for the event buffers to be written to file if they are full. This can degrade database performance, as the suspended agent and any dependent agents cannot run until the buffers are clear. Use the BLOCKED clause to ensure no losses of event data:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
BLOCKED
```

Event monitors are blocked by default. If database performance is of greater importance than collecting every single event record, use non-blocked event monitors. In this case, each agent that generates an event will not wait for the event buffers to be written to file if they are full. As a result, non-blocked event monitors are subject to data loss on highly active systems. Use the NONBLOCKED clause to minimize the additional processing time caused by event monitoring:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
NONBLOCKED
```

5. Specify the maximum number of event files that can be collected for an event monitor. If this limit is reached, the event monitor will deactivate itself.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
NONBLOCKED MAXFILES 5
```

5 is the maximum number of event files that will be created.

You can also specify that there is no limit to the number of event files that the event monitor can create:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
NONBLOCKED MAXFILES NONE
```

6. Specify the maximum size (in 4K pages) for each event file created by an event monitor. If this limit is reached, a new file is created.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
NONBLOCKED MAXFILES 5 MAXFILESIZE 32
```

32 is the maximum number of 4K pages that an event file can contain.

This value must be greater than the value specified by the `BUFFERSIZE` parameter. You can also specify that there is to be no limit on an event file's size:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
      NONBLOCKED MAXFILES NONE MAXFILESIZE NONE
```

7. Specify if the event monitor is to be activated automatically each time the database starts. By default, event monitors (with the exception of the WLM event monitors) are not activated automatically when the database starts.
  - To create an event monitor that starts automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
      NONBLOCKED AUTOSTART
```
  - To create an event monitor that does not start automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
      NONBLOCKED MANUALSTART
```
8. To activate or deactivate an event monitor, use the `SET EVENT MONITOR STATE` statement.

## Results

Once a file event monitor is created and activated, it will record monitoring data as its specified events occur.

### Event monitor file management:

With some event monitors, you can have event data written to text files. You can configure some upper bounds on the number of files created, as well as their size with options on the `CREATE` or `ALTER EVENT MONITOR` statements.

A file event monitor enables the event monitor to store its event records in files. All the output of the event monitor goes in the directory supplied in the `FILE` parameter for the `CREATE EVENT MONITOR` statement. Before the monitor is activated, the directory must exist, or the `SET EVENT MONITOR` command will return an error; the directory will not be created by the database manager if it does not already exist.

**Important:** When a file event monitor is first activated, a control file named `db2event.ct1` is created in this directory. Do not remove or modify this file.

By default, an event monitor writes its trace to a single file, called `00000000.evt`. This file keeps growing as long as there is space on the file system. If you specified a file size limit with the `MAXFILESIZE` parameter of the `CREATE EVENT MONITOR` statement, then when a file is full, output is directed to a new file. The number that makes up the file name is increased by 1 each time a new file is created. Hence, the active file is the file with the highest number.

You can limit the maximum size of the entire event monitor trace by also using the `MAXFILES` parameter of the `CREATE EVENT MONITOR` statement. When the

number of files reaches the maximum defined by **MAXFILES**, the event monitor deactivates itself and the following message is written to the administration notification log.

```
DIA1601I Event Monitor monitor-name was deactivated when it reached
its preset MAXFILES and MAXFILESIZE limit.
```

If you receive this message, do not delete any of the event monitor files. If you do, you will not be able to view any of the event monitor information (even that contained in any remaining files) using the **db2evmon** command. Instead take one of the following actions:

- Recreate the event monitor without the **MAXFILES** and **MAXFILESIZE** limits.
- Leave the limits imposed by the **MAXFILES** and **MAXFILESIZE** parameters in place, but move all but the most recent \*.evt files in the directory to another directory or file system. You can then view the event monitor information from the files in the new directory. You can create a script to do this automatically if you want.

Either way, you must reactivate the event monitor using the statement **SET EVENT MONITOR *event-monitor-name* STATE 1** to start collecting information again after you receive the DIA1601I message.

When a file event monitor is restarted, it can either erase any existing data or append new data to it. This option is specified in the **CREATE EVENT MONITOR** statement, where either an **APPEND** monitor or a **REPLACE** monitor can be created. **APPEND** is the default option. An **APPEND** event monitor starts writing at the end of the file it was last using. If you have removed that file, the next file number in sequence is used. When an append event monitor is restarted, only a `start_event` is generated. The event log header and database header are generated only for the first activation. A **REPLACE** event monitor always deletes existing event files and starts writing at `00000000.evt`.

**Note:** If you did not use the **REPLACE** option for the event monitor, you can perform the following steps to force the event monitor to start collecting a new set of data:

1. Deactivate the event monitor using the **SET EVENT MONITOR *event-monitor-name* STATE 0** command.
2. Delete all files in the directory that was specified by the **FILE** option of the **CREATE EVENT MONITOR** statement.
3. Reactivate the event monitor using the **SET EVENT MONITOR *event-monitor-name* STATE 1** command.

If a file event monitor runs out of disk space, it shuts itself down after logging a system-error-level message in the administration notification log.

You might want to process monitor data while the event monitor is active. This is possible, and furthermore, when you are finished processing a file, you can delete it, freeing up space for further monitoring data. An event monitor cannot be forced to switch to the next file unless you stop and restart it. It must also be in **APPEND** mode. To track which events have been processed in the active file, you can create an application that simply tracks the file number and location of the last record processed. When processing the trace the next time around, the application can seek to that file location.

## Creating a pipe event monitor

When creating an event monitor you must determine where the information it collects is to be stored. A pipe event monitor streams event records directly from the event monitor, to a named pipe.

### Before you begin

- You need SQLADM or DBADM authority to create a pipe event monitor.
- This task assumes the named pipe is already created. To create a named pipe on UNIX or Linux systems, use the **mkfifo** command provided on those systems.

### About this task

It is the responsibility of the monitoring application to promptly read the data from the pipe as the event monitor writes the event data. If the event monitor is unable to write data to the pipe (for example, if it is full), monitor data will be lost.

Pipe event monitors are defined with the CREATE EVENT MONITOR statement.

### Procedure

1. Indicate that event monitor data is to be directed to a named pipe.

```
CREATE EVENT MONITOR myevmon FOR eventtype
                        WRITE TO PIPE '/home/dbadmin/dlevents'
```

myevmon is the name of the event monitor.

/home/dbadmin/dlevents is the name of the named pipe (on UNIX) to where the event monitor will direct the event records. The CREATE EVENT MONITOR statement supports UNIX and Windows pipe naming syntax.

The named pipe specified in the CREATE EVENT MONITOR statement must be present and open when you activate the event monitor. If you specify that the event monitor is to start automatically, the named pipe must exist before the event monitor's creation.

2. Specify the types of events to be monitored. You can monitor one or more event types with a single event monitor.

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                        WRITE TO PIPE '/home/dbadmin/myevents'
```

This event monitor will monitor for the BUFFERPOOLS and TABLESPACES event types.

3. Specify if the event monitor is to be activated automatically each time the database starts. By default, event monitors are not activated automatically when the database starts.

- To create an event monitor that starts automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                        WRITE TO PIPE '/home/dbadmin/myevents'
                        AUTOSTART
```

- To create an event monitor that does not start automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                        WRITE TO PIPE '/home/dbadmin/myevents'
                        MANUALSTART
```

4. Start the client application that reads from the named pipe. For example, you can start the db2evmon tool to process the data as it is delivered to the pipe.

5. To activate or deactivate an event monitor, use the SET EVENT MONITOR STATE statement.

## Results

After a pipe event monitor is created and activated, it will record monitoring data as its specified events occur.

### Event monitor named pipe management:

With some event monitors, you can have event data written to named pipes. What follows are some guidelines on how to use named pipe event monitors more effectively.

A pipe event monitor enables the processing of the event monitor data stream through a named pipe. Using a pipe event monitor is desirable if you need to process event records in real time. Another important advantage is that your application can ignore unwanted data as it is read off the pipe, giving the opportunity to considerably reduce storage requirements.

On AIX<sup>®</sup>, you can create named pipes by using the mkfifo command. On Linux and other UNIX types (such as the Solaris operating system) use the pipe() routine. On Windows, you can create named pipes by using the CreateNamedPipe() routine.

When you direct data to a pipe, I/O is always blocked and the only buffering is that performed by the pipe. It is the responsibility of the monitoring application to promptly read the data from the pipe as the event monitor writes the event data. If the event monitor is unable to write the data to the pipe (for example, because the pipe is full), monitor data will be lost.

In addition, there must be enough space in the named pipe to handle incoming event records. If the application does not read the data from the named pipe fast enough, the pipe will fill up and overflow. The smaller the pipe buffer, the greater the chance of an overflow.

When a pipe overflow occurs, the monitor creates overflow event records indicating that an overflow has occurred. The event monitor is not turned off, but monitor data is lost. If there are outstanding overflow event records when the monitor is deactivated, a diagnostic message will be logged. Otherwise, the overflow event records will be written to the pipe when possible.

The amount of data that can be written to a pipe at any one time is determined by the underlying operating system. If your operating system allows you to define the size of the pipe buffer, use a pipe buffer of at least 32K. For high-volume event monitors, you should set the monitoring application's process priority equal to or higher than the agent process priority.

It is possible for the data stream coming from a single write operation of an activities or statistics event monitor to contain more data than can be written to the named pipe. In these situations, the data stream is split into blocks that can fit into the buffer, and each block is identified with a header: The first block is identified by a logical header with the element ID SQLM\_ELM\_EVENT\_STARTPIPEBLOCK. The last block is identified by a logical header with element ID SQLM\_ELM\_EVENT\_ENDPIPEBLOCK. All blocks in between are identified by logical headers with element ID SQLM\_ELM\_EVENT\_MIDPIPEBLOCK. The

monitoring application that is reading the pipe must be aware of these headers, and reassemble the blocks back into the complete data stream, stripping off the block headers as needed and reassembling the blocks to form a complete, valid data stream. The db2evmon tool provides this capability; it provides formatted output for all events generated by an event monitor that writes to a named pipe, reassembling the blocks as needed. If you want to process only selected events or monitor elements, you can write your own application to do so.

### Write-to-table and file event monitor buffering

For some write-to-table and file event monitors, the event monitor stores output in a buffer before writing it to a file or table. Table 11 shows which event monitors use such output buffers.

Table 11. Event monitors and output buffers

| Event monitor type       | Writes output to buffers before writing to disk? |
|--------------------------|--|
| Activities               | No   |
| Bufferpools              | Yes  |
| Change history           | No   |
| Connections              | Yes  |
| Database                 | Yes  |
| Deadlocks (all versions) | Yes  |
| Locking                  | No   |
| Package cache            | No   |
| Statements               | Yes  |
| Statistics               | Yes  |
| Tablespaces              | Yes  |
| Tables                   | Yes  |
| Transactions             | Yes  |
| Unit of work             | No   |

Event monitors that do not use buffers use a newer, faster mechanism for writing output to disk, eliminating the need for buffers.

For those event monitors that use buffers, records are written to disk automatically when a buffer is full. Therefore, you can improve monitoring performance for event monitors with high amounts of throughput by specifying larger buffers to reduce the number of disk accesses. To force an event monitor to flush its buffers, you must either deactivate it or empty the buffers by using the FLUSH EVENT MONITOR statement.

Event monitors that use buffers let you specify whether the event monitor output is to be *blocked* or *non-blocked*. A blocked event monitor suspends the database process that is sending monitor data when both of its buffers are full. This is to ensure that no event records are discarded while the blocked event monitor is active. The suspended database process and consequently, any dependent database processes cannot run until a buffer has been written. This can introduce a significant performance consumption, depending on the type of workload and the speed of the I/O device. Event monitors are blocked by default.

A non-blocked event monitor discards monitor data coming from agents when data is coming faster than the event monitor can write the data. This prevents event monitoring from becoming a performance burden on other database activities.

An event monitor that has discarded event records generates an overflow event. It specifies the start and stop time during which the monitor was discarding events and the number of events that were discarded during that period. It is possible for an event monitor to terminate or be deactivated with a pending overflow to report. If this occurs, the following message is written to the admin log:

DIA2503I Event Monitor monitor-name had a pending overflow record when it was deactivated.

Loss of event monitoring data can also occur for individual event records. If the length of an event record exceeds the event buffer size, the data that does not fit in the buffer is truncated. For example, this situation might occur if you are capturing the `stmt_text` monitor element and applications attached to the database being monitored issue lengthy SQL statements. If you must capture all of the event record information, specify larger buffers. Keep in mind that larger buffers result in less frequent writes to file or table.

### Event monitor self-describing data stream

An event monitor that writes to a pipe or file outputs a binary stream of logical data groupings that are exactly the same for both pipe and file event monitors. You can format the data stream either by using the `db2evmon` command or by developing a client application. This data stream is presented in a self-describing format. Figure 4 shows the structure of the data stream and Table 12 on page 109 provides some examples of the logical data groups and monitor elements that could be returned.

**Note:** In the examples and tables descriptive names are used for the identifiers. These names are prefixed by `SQLM_ELM_` in the actual data stream. For example, `db_event` would appear as `SQLM_ELM_DB_EVENT` in the event monitor output. Types are prefixed with `SQLM_TYPE_` in the actual data stream. For example, headers appear as `SQLM_TYPE_HEADER` in the data stream.

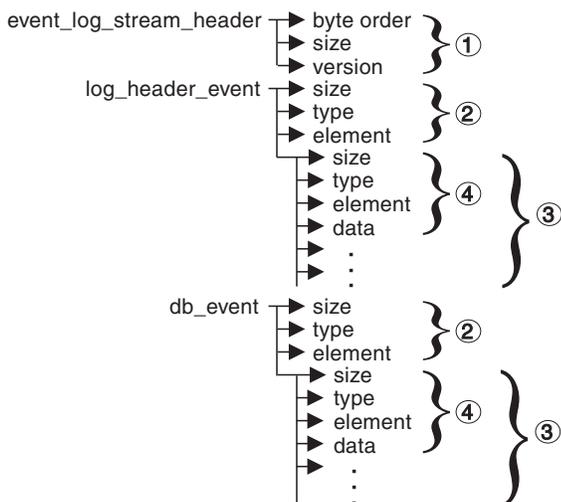


Figure 4. Pipe or File Event Monitor Data Stream

1. The structure of the `sqlm_event_log_data_stream_header` is different than the other headers in the data stream. The version field determines if the output can be processed as a self-describing data stream.

This header has the same size and type as pre-Version 6 event monitor streams. This allows applications to determine if the event monitor output is self-describing or is in the pre-Version 6 static format.

**Note:** This monitor element is extracted by reading `sizeof(sqlm_event_log_data_stream)` bytes from the data stream.

2. Each logical data group begins with a header that indicates its size and element name. This does not apply `event_log_stream_header`, as its size element contains a dummy value to maintain backwards compatibility.
3. The size element in the header indicates the size of all the data in that logical data group.
4. Monitor element information follows its logical data group header and is also self-describing.

Table 12. Sample event data stream

| Logical Data Group                   | Data Stream  | Description  |
|--------------------------------------|--|--|
| <code>event_log_stream_header</code> | <pre> └─▶sqlm_little_endian └─▶200 └─▶sqlm_dbmon_version9 </pre>   | <p>Not used (for compatibility with previous releases).</p> <p>Not used (for compatibility with previous releases).</p> <p>The version of the database manager that returned the data. Event monitors write data in the self-describing format.</p>  |
| <code>log_header_event</code>        | <pre> └─▶100 └─▶header └─▶log_header └─▶4 └─▶u32bit └─▶byte_order └─▶little_endian └─▶2 └─▶u16bit └─▶codepage_id └─▶850 </pre> | <p>Size of the logical data group.</p> <p>Indicates the start of a logical data group.</p> <p>Name of the logical data group.</p> <p>Size of the data stored in this monitor element.</p> <p>Monitor element type - 32 bit numeric.</p> <p>The name of the monitor element collected.</p> <p>The collected value for this element.</p> <p>Size of the data stored in this monitor element.</p> <p>Monitor element type - unsigned 16 bit numeric.</p> <p>The name of the monitor element collected.</p> <p>The collected value for this element.</p> |
| <code>db_event</code>                | <pre> └─▶100 └─▶header └─▶db_event └─▶4 └─▶u32bit └─▶lock_waits └─▶2 </pre>  | <p>Size of the logical data group.</p> <p>Indicates the start of a logical data group.</p> <p>Name of the logical data group.</p> <p>Size of the data stored in this monitor element</p> <p>Monitor element type - unsigned 32 bit numeric.</p> <p>The name of the monitor element collected.</p> <p>The collected value for this element.</p>   |

The `event_log_stream_header` identifies the version of the database manager that returned the data. Event monitors write their data in the self-describing format. An event monitor, unlike a snapshot monitor, does not have a **size** element that returns the total size of the trace. The number present in `event_log_stream_header` is a dummy value present for backwards compatibility. The total size of an event trace is not known when the `event_log_stream_header` is written. You typically read an event monitor trace until you reach an end of file or pipe.

The log header describes the characteristics of the trace, containing information such as the memory model (for example little endian) of the server where the trace

was collected, and the code page of the database. You might have to do byte swapping on numeric values, if the system where you read the trace has a different memory model than the server (for example, if you are reading a trace from a UNIX server on a Windows 2000 system). Code page translation might also need to be done if the database is configured in a different language than the machine from which you read the trace. When reading the trace, you can use the **size** element to skip a logical data group in the trace.

### Event type mappings to logical data groups

For file and pipe event monitors, event monitor output consists of an ordered series of logical data groupings. Regardless of the event monitor type, the output records always contain the same starting logical data groups. These frame the logical data groups whose presence varies depending on the event types recorded by the event monitor.

For file and pipe event monitors, event records may be generated for any connection and may therefore appear in mixed order in the stream. This means that you may get a transaction event for Connection 1, immediately followed by a connection event for Connection 2. However, records belonging to a single connection or a single event will appear in their logical order. For example, a statement record (end of statement) always precedes a transaction record (end of UOW), if any. Similarly, a deadlock event record always precedes the deadlocked connection event records for each connection involved in the deadlock. The **application id** or **application handle (agent\_id)** can be used to match records with a connection.

Connection header events are normally written for each connection to the database. For deadlocks with details event monitors, they are only written when the deadlock occurs. In this case, connection header events are only written for participants in the deadlock and not for all connections to the database.

The logical data groupings are ordered according to four different levels: Monitor, Prolog, Contents, and Epilog. Following are detailed descriptions for each level, including the corresponding event types and logical data groups.

### Monitor

Information at the Monitor level is generated for all event monitors. It consists of event monitor metadata.

*Table 13. Event Monitor Data Stream: Monitor Section*

| Event type    | Logical data group      | Available information  |
|---------------|-------------------------|--|
| Monitor Level | event_log_stream_header | Identifies the version level and byte order of the event monitor. Applications can use this header to determine whether they can handle the evmon output stream. |

## Prolog

The Prolog information is generated when the event monitor is activated.

*Table 14. Event Monitor Data Stream: Prolog Section*

| Event type          | Logical data group | Available information  |
|---------------------|--------------------|--|
| Log Header          | event_log_header   | Characteristics of the trace, for example server type and memory layout.   |
| Database Header     | event_db_header    | Database name, path and activation time.   |
| Event Monitor Start | event_start        | Time when the monitor was started or restarted.  |
| Connection Header   | event_connheader   | One for each current connection, includes connection time and application name. Event connection headers are only generated for connection, statement, transaction, and deadlock event monitors. Deadlocks with details event monitors produce connection headers only when a deadlock occurs. |

## Contents

Information specific to the event monitor's specified event types is presented in the Contents section.

*Table 15. Event Monitor Data Stream: Contents Section*

| Event type                               | Logical data group          | Available information   |
|--|-----------------------------|---|
| Statement Event                          | event_stmt                  | Statement level data, including text for dynamic statements. Statement event monitors do not log fetches.   |
| Subsection Event                         | event_subsection            | Subsection level data.  |
| Transaction Event <sup>1</sup>           | event_xact                  | Transaction level data.   |
| Connection Event                         | event_conn                  | Connection level data.  |
| Deadlock Event                           | event_deadlock              | Deadlock level data.  |
| Deadlocked Connection Event              | event_dlconn                | One for each connection involved in the deadlock, includes applications involved and locks in contention.   |
| Deadlocked Connection Event with Details | event_detailed_dlconn, lock | One for each connection involved in the deadlock, includes applications involved, locks in contention, current statement information, and other locks held by the application contention. |
| Overflow                                 | event_overflow              | Number of records lost - generated when writer cannot keep up with a (non-blocked) event monitor.   |

Table 15. Event Monitor Data Stream: Contents Section (continued)

| Event type   | Logical data group        | Available information   |
|--|---------------------------|---|
| Deadlocks with details history <sup>2</sup>        | event_stmt_history        | List of statements executed in any unit of work that was involved in a deadlock.  |
| Deadlocks with details history values <sup>2</sup> | event_data_value          | Parameter markers for a statement in the event_stmt_history list.   |
| Activities   | event_activity            | List of activities that completed executing on the system or were captured before completion.   |
|  | event_activystmt          | Information about the statement the activity was executing if the activity type was a statement.  |
|  | event_activityvals        | The data values used as input variables for each activity that is an SQL statement. These data values do not include LOB data, long data, or structured type data.                |
| Statistics   | event_scstats             | Statistics computed from the activities that executed within each service class, work class, or workload in the system, as well as statistics computed from the threshold queues. |
|  | event_wcstats             |   |
|  | event_wlstats             |   |
|  | event_qstats              |   |
|  | event_histogrambin        |   |
| Threshold violations                               | event_thresholdviolations | Information identifying the threshold violated and the time of violation.   |

<sup>1</sup> This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR UNIT OF WORK statement to monitor transaction events.

<sup>2</sup> This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Epilog

The Epilog information is generated during database deactivation (last application finished disconnecting):

Table 16. Event Monitor Data Stream: Epilog Section

| Event type        | Logical data group | Available information        |
|-------------------|--------------------|------------------------------|
| Database Event    | event_db           | Database manager level data. |
| Buffer Pool Event | event_bufferpool   | Buffer pool level data.      |
| Table Space Event | event_tablespace   | Table space level data.      |
| Table Event       | event_table        | Table level data.            |

## Displaying a list of event monitors created in your database

You can see what event monitors are already defined in your database by using the catalog view SYSCAT.EVENTMONITORS.

### Procedure

To view a list of the event monitors that you defined on your system, query the catalog view SYSCAT.EVENTMONITORS. For example, to see a list of event monitors that includes the event monitor name, the target output type (that is, a regular table, file, named pipe, or unformatted event table), and the owner, you can use a query such as the following one:

```
SELECT SUBSTR(EVMONNAME,1,20) AS EVMON_NAME, TARGET_TYPE, OWNER
FROM SYSCAT.EVENTMONITORS
```

The preceding query returns results similar to those that follow:

| EVMON_NAME        | TARGET_TYPE | OWNER    |
|-------------------|-------------|----------|
| DB2DETAILDEADLOCK | F           | DBADMIN1 |
| CACHEEVMON        | T           | DBADMIN1 |
| INVTLOCK          | T           | DBADMIN1 |
| INVTUOW           | T           | DBADMIN1 |
| INVTACT           | T           | DBADMIN1 |
| INVTSTATS         | T           | DBADMIN1 |
| INVTTHRESHOLD     | T           | DBADMIN1 |
| TABLE_INVTTABLE   | T           | DBADMIN1 |
| BUFFER_INV        | T           | DBADMIN1 |
| TABLESPACES_INV   | T           | DBADMIN1 |
| CONNECTIONS_INV   | T           | DBADMIN1 |
| TRANSAC_INV       | T           | DBADMIN1 |
| DEADLOCK_INV      | T           | DBADMIN1 |
| QUINN_JN_LOC_UNF  | U           | DBADMIN1 |
| UNFORM            | U           | DBADMIN1 |
| RM                | U           | DBADMIN1 |
| UOWINV            | U           | DBADMIN1 |
| LOCK_UP_STAFF     | U           | DBADMIN1 |
| INVTLOCK2         | T           | DBADMIN1 |
| STAFF_UOW         | T           | DBADMIN1 |
| STAFFSTATS        | T           | DBADMIN1 |

21 record(s) selected.

### Examples

You can also use a catalog view to see which event monitors exist for monitoring a specific type of event. The SYSCAT.EVENTS view returns a list of event monitors and the type of events for which they record data.

```
SELECT SUBSTR(TYPE,1,20) AS EVENT_TYPE,
       SUBSTR(EVMONNAME,1,20) AS EVENT_MONITOR_NAME
FROM SYSCAT.EVENTS
ORDER BY TYPE
```

| EVENT_TYPE      | EVENT_MONITOR_NAME |
|-----------------|--------------------|
| ACTIVITIES      | INVTACT            |
| BUFFERPOOLS     | BUFFER_INV         |
| CONNECTIONS     | CONNECTIONS_INV    |
| DEADLOCKS       | DEADLOCK_INV       |
| DETAILDEADLOCKS | DB2DETAILDEADLOCK  |
| LOCKING         | INVTLOCK           |
| LOCKING         | QUINN_JN_LOC_UNF   |
| LOCKING         | UNFORM             |
| LOCKING         | RM                 |

|                     |                 |
|---------------------|-----------------|
| LOCKING             | LOCK_UP_STAFF   |
| LOCKING             | INVTLOCK2       |
| PKGCACHEBASE        | CACHEEVMON      |
| STATISTICS          | INVTSTATS       |
| STATISTICS          | STAFFSTATS      |
| TABLES              | TABLE_INVTTABLE |
| TABLESPACES         | TABLESPACES_INV |
| THRESHOLDVIOLATIONS | INVTTHRESHOLD   |
| TRANSACTIONS        | TRANSAC_INV     |
| UOW                 | INVTUOW         |
| UOW                 | UOWINVT         |
| UOW                 | STAFF_UOW       |

21 record(s) selected.

## Event monitors for partitioned databases and databases in a DB2 pureScale environment

Generally, event monitors on partitioned database systems or in a DB2 pureScale<sup>®</sup> environment work similarly to event monitors that run on nonpartitioned, single-member databases. However, there are some differences to be aware of.

### Partitioned database environments

#### Event monitors that write to regular tables and unformatted event (UE) tables

You cannot create event monitors that write to regular tables and UE tables on a specific partition. Instead, for a partitioned database environment, an event monitor process runs on each of the partitions. More specifically, the event monitor process runs on the members for each partition that belong to the database partition groups in which the target tables exist.

Each partition where the event monitor process runs has the same set of target tables for a specific event monitor. The data in these tables is different from partition to partition because the data for a specific partition reflects only events that take place on that partition. For table event monitors, you can retrieve aggregate values from all the partitions by issuing SQL statements to collect data from event monitor tables from each partition. For UE table event monitors, you can aggregate data across partitions by using the SQL statement that you specify for the `EVMON_FORMAT_UE_TO_TABLE` stored procedure or by using the `EVMON_FORMAT_UE_TO_XML` table function.

The first column of each event monitor table is named `PARTITION_KEY` and is used as the partitioning key for the table. The value of this column is chosen so that each event monitor process inserts data into the database partition on which the process is running. That is, insert operations are performed locally on the database partition where the event monitor process is running. On any database partition, the `PARTITION_KEY` field contains the same value. As a result, if you drop a data partition and data redistribution is performed, all data on the dropped database partition goes to one other database partition instead of being evenly distributed. Therefore, before dropping a database partition, consider deleting all table rows on that database partition.

In addition, in partitioned database environments, a column named `PARTITION_NUMBER`, or `MEMBER` is defined for each table. This column contains the number of the partition or member on which the data was inserted.

Events are written to the event monitor target tables on those partitions where the table space for the target tables exists. If the table space for the event monitor target tables does not exist on any partition where the event monitor runs, no data is collected on those partitions, and no error is returned. Moreover, no log records for these events are written where the table space does not exist. This behavior means that you can choose a subset of partitions for monitoring by creating a table space that exists only on certain partitions.

During write-to-table event monitor activation, the CONTROL table rows for FIRST\_CONNECT and EVMON\_START are inserted on all database partitions where the table space for target tables exists.

If a partition is not yet active when an event monitor is activated, the event monitor is activated when that partition is next activated.

### Event monitors that write to files and named pipes

File and pipe event monitors, with one exception, capture only events that take place on the database partition on which they are running (the *monitor partition*). Such an event monitor is known as a *local event monitor*. The exception is the DEADLOCK event monitor; you can create it as a local or a *global* event monitor. When you create it as a global event monitor, deadlock information is collected on all database partitions and is reported to the specific database partition where the event monitor process runs.<sup>3</sup>

When you create a file or pipe event monitor in a partitioned database environment, you can specify the partition that you want it to run on as part of the CREATE EVENT MONITOR statement. If you omit the partition number, the event monitor runs on the database partition that was connected when you created the event monitor.

An event monitor can be activated only if the monitor partition is active. If you use the SET EVENT MONITOR statement to activate an event monitor but the monitor partition is not yet active, event monitor activation occurs when the monitor partition is next started. Furthermore, the event monitor is activated automatically until you explicitly deactivate the event monitor or the instance. For example, consider the following sequence of statements:

```
DB2 CONNECT TO PAYROLL
DB2 CREATE EVENT MONITOR ABC ... ON DBPARTITIONNUM 2
DB2 SET EVENT MONITOR ABC STATE 1
```

After these statements are run, event monitor ABC activates automatically whenever the database PAYROLL is activated on database partition 2. This automatic activation occurs until the statement DB2 SET EVENT MONITOR ABC STATE 0 is issued or partition 2 is stopped.

If you add database partitions, the existing global, table, or UE table event monitors do not automatically start collecting data for the newly created partitions. To collect and record data about the new partitions, you must take one of the following steps:

- For global event monitors (that is, a DEADLOCKS event monitor), restart the event monitors.

---

3. This event monitor is deprecated. The LOCKING event monitor is the preferred event monitor for capturing lock and deadlock event information.

- For table or UE table event monitors, drop, re-create, and restart the event monitors.

## DB2 pureScale environments

In DB2 pureScale environments, there is effectively one data partition, with two or more members that process data. Thus, when you create an event monitor, event monitor processes run on all members, regardless of whether they write to a file, pipe, tables, or a UE table.

Event data is reported on a per-member basis. As a result, monitor elements or metrics that are associated with a member, such as the **total\_cpu\_time** monitor element, report data that is specific to that member. However, other monitor elements related to the data itself, such as the **tablespace\_total\_pages** monitor element, reflect the same values regardless of what member reports them.

## Examples

### Example 1: Creating a write-to-file event monitor in a partitioned database environment

The example that follows shows how to create an event monitor that runs and collects data for buffer pool-related events on partition 3, writing its output to a file:

```
CREATE EVENT MONITOR bpmon FOR BUFFERPOOLS
    WRITE TO FILE '/tmp/dlevents'
    ON DBPARTITION 3
```

### Example 2: Creating a table event monitor in a partitioned database environment

The example that follows shows how to create a table monitor that runs and collects data for activities-related events and writes its output to a table:

```
CREATE EVENT MONITOR myacts FOR ACTIVITIES
    WRITE TO TABLE
```

In this example, because no logical data groups are specified for the event monitor, tables are created for all logical data groups associated with this type of event monitor. Each of these tables is created on each partition in the default table space if the default table space exists on each partition. The data that is collected in the tables on each database partition pertains to events that take place on that partition.

To view event monitor data from selected partitions, issue a **SELECT** statement that queries those partitions:

```
SELECT TOTAL_CPU_TIME FROM myacts WHERE PARTITION_NUMBER = 3
```

## Enabling event monitor data collection

Depending on the type of event monitor you are using, you might need to configure collection after you create the event monitor. By default, some event monitors collect certain data immediately when activated. Other event monitors require that you explicitly configure data collection independently of creating the event monitor. These types of event monitors are sometimes referred to as *passive* event monitors.

## Before you begin

All event monitors must be activated before any data is written its target output table or tables (regular or UE), file or pipe. Some event monitors are configured by default as AUTOSTART event monitors. This means they are activated automatically when the database is activated. Others are configured by default to require that you activate them manually. Either way, you can override the default startup options. However, to start an automatic event monitor after you create it, but before the next database activation, you must use the SET EVENT MONITOR STATE statement to activate it manually.

## About this task

Some event monitors support the use of a WHERE clause on the CREATE or ALTER EVENT MONITOR statement to capture event information selectively. The following event monitors, however, provide the ability to control what event data is collected independently of the event monitor definition:

- Activities
- Change history
- Locking
- Statistics
- Unit of work

Some of the event monitors listed collect certain types of data by default after the event monitor is activated; others require that you explicitly enable data collection. Either way, you can enable data collection in one of two ways, depending on the scope of activities for which you want data collected:

### All activities in the database

To have monitor data collected across all activities in the database, you modify the appropriate configuration parameter for the type of data you are interested in. For example, to have unit of work data collected for all units of work that run in the database, set `mon_uow_data` to BASE. In some cases, the default settings for configuration parameters are such that some type of data is always collected if there is an appropriate event monitor active to receive the data. For example, the default setting for `mon_req_metrics` is BASE; unless you override this setting, any active statistics or unit of work event monitor will record the values for the BASE set of request monitor elements.

**Remember:** Event monitors that support the use of the WHERE predicate collect only the data that satisfies the conditions specified in that predicate, regardless of the settings for any relevant configuration parameters.

### Selected activities

Some event monitors - in particular, the workload management event monitors (threshold violations, statistics and activities) - provide the ability to control data collection for specific workload management objects. For example, you might choose to collect activity information for activities running in a specific service superclass. Configuring collection at this level generally involves adding a COLLECT clause to the CREATE or ALTER WORKLOAD (or SERVICE CLASS or WORK ACTION) statements to specify what type of information to collect for activities running under the auspices of that WLM object. For example, to enable the collection of extended statistics information for the service class urgent, you might use the following statement:

```
ALTER SERVICE CLASS urgent
  COLLECT AGGREGATE ACTIVITY DATA EXTENDED
```

**Note:** If a COLLECT clause is specified in a WLM CREATE or ALTER statement, the settings specified in the clause take precedence for that WLM object over any database-wide setting configured using a configuration parameter. For example, if `mon_req_metrics` is set to EXTENDED, and if workload payroll was configured to collect BASErequest metrics (for example, CREATE WORKLOAD payroll COLLECT REQUEST METRICS BASE), then extended request metrics are collected for all activities in the database *except* for the payroll workload.

## Procedure

To enable collection of data for one of the types of event monitors shown at the beginning of this section, perform the following steps:

1. Determine what, if any data is already collected by default. The data you are interested in might be collected without you having to change any settings.
2. Decide on the scope of activities for which you want to collect data. Do you want to collect data for the entire database, or only for specific workloads, service class or work actions?
3. Decide what types of monitor elements you want to collect. Some event monitors support the collection of different types of monitor data, such as request monitor elements, activity data, and so on.
4. For the different sets of monitor data collected, decide the scope of data to be collected within each set. You generally have the choice of collecting no data (NONE), basic data (BASE), or extended data (EXTENDED). See to determine what data is collected for each setting.
5. Based on the decisions made in the preceding steps, configure data collection using either a configuration parameter or a COLLECT clause.
  - a. To configure collection across the entire database, set the appropriate configuration parameter. For example, to enable the collection of lock wait information with history by the locking event monitor on the database SALES, run the following command.
  - b. To configure collection for a specific workload, create or modify the workload, including the appropriate COLLECT clause. For example, to configure the collection of lock wait data with statement history for locks waiting longer than 5 seconds in the MANAGERS workload, run a statement like the one that follows:

```
UPDATE DATABASE CONFIGURATION for SALES USING mon_lockwait HISTORY
```

```
ALTER WORKLOAD MANAGERS
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 5 SECONDS
  WITH HISTORY
```

## What to do next

Now that the event monitor is created and active, and data collection is enabled, run your applications or workload.

## Methods for accessing event monitor information

Depending on the type of event monitor that you are using and the type of output it generates, there are different options for accessing and viewing event monitor data.

For example:

- Data produced by table event monitors can be queried directly using SQL.
- Data from event monitors that write to pipes can be viewed as it is produced.
- Data from file event monitors can be viewed by opening the output file after the event monitor is deactivated.
- Data from both file and pipe event monitors can also be formatted into a report using the **db2evmon** command.
- Data written to UE tables must be post-processed before it can be examined. UE event monitor data can be converted to tables or to XML, which makes it possible to query the data using SQL or XML query techniques. Alternatively, you can format the data in a UE table into a formatted report without going through a conversion process.

The sections that follow describe the different ways you can access information produced by event monitors.

## Accessing event monitor data in regular tables

You can use SQL to directly access event monitor data that is written to regular relational tables.

### Before you begin

Before accessing data, you must perform the following tasks:

- Create and activate the event monitor
- Enable data collection if required for the type of event monitor that you are using and the type of data that you want to collect
- Run the workload or applications for which you want to collect monitoring data

Optionally, depending on how you are using the event monitor data, deactivate data collection before you start examining the event data. If the event monitor remains active, it continues to write data to the output tables. Therefore, the results from one query might differ from the results that you obtain by running the same query later on.

### About this task

Accessing event monitor data from relational tables involves using SQL to formulate queries to retrieve data from the tables produced by the event monitor.

### Procedure

To retrieve information from the tables that are produced by an event monitor that writes to tables:

1. Formulate a SELECT statement to display the monitor element data you want to see. For example, to request lock data for the payroll workload from a locking event monitor named mylocks, you might use a query such as the following one:

```
SELECT DISTINCT CAST(STMT_TEXT AS VARCHAR(25)) STMT, LP.PARTICIPANT_NO,
  VARCHAR(LP.APPL_NAME,10) APPL_NAME, LP.LOCK_MODE_REQUESTED,
  LP.PARTICIPANT_TYPE
FROM LOCK_PARTICIPANT_ACTIVITIES_LOCK_MYLOCKS AS LPA
JOIN LOCK_PARTICIPANTS_LOCK_MYLOCKS AS LP
ON LPA.EVENT_ID = LP.EVENT_ID
WHERE LP.WORKLOAD_NAME = 'PAYROLL'
```

In this example, data from the LOCK\_PARTICIPANTS table from the event monitor mylocks is joined with information from the LOCK\_PARTICIPANTS\_ACTIVITIES table to return the following results.

2. Run the SQL statement.

### Results

```

STMT                                PARTICIPANT_NO  APPL_NAME  LOCK_WAIT_VAL
-----
select * from staff                 2 db2bp    0
select * from staff                 1 db2bp    1000

LOCK_MODE_REQUESTED  PARTICIPANT_TYPE
-----
0 OWNER
1 REQUESTER

```

2 record(s) selected.

### Methods for accessing information in unformatted event tables

There are different ways to access the information in unformatted event (UE) tables. You can generate a text report intended to be read. Alternatively, you can extract the data into relational tables or XML; this approach lets you query the data using SQL or pureXML®.

Event monitors that write to UE tables write event data in a binary format. You can access this data using the db2evmonfmt command or routines provided for this purpose.

With the db2evmonfmt command you can:

- select events of interest based on the following attributes: event ID, event type, time period, application, workload, or service class.
- choose whether to receive the output in the form of a text report or a formatted XML document.
- completely control the output format by creating your own XSLT style sheets instead of using the ones provided with db2evmonfmt.

You can also extract data from an unformatted event table using the following routines:

- EVMON\_FORMAT\_UE\_TO\_XML - extracts data from an unformatted event table into an XML document.
- EVMON\_FORMAT\_UE\_TO\_TABLES - extracts data from an unformatted event table into a set of relational tables.

With these two routines, you can use a SELECT statement to specify the exact rows from the unformatted event table that you want to extract.

### db2evmonfmt tool for reading event monitor data:

The Java-based, generic XML parser tool, **db2evmonfmt**, produces a readable flat-text output (text version) or a formatted XML output from the data generated by an event monitor that uses the unformatted event table. Based on the parameters that you specify, the **db2evmonfmt** tool determines how to parse the event monitor data and the type of output to create.

The **db2evmonfmt** tool is provided as Java source code. You must setup and compile this tool, before you can use it, by performing the following steps:

1. Locate the source code in the `sqllib/samples/java/jdbc` directory
2. Follow the instructions embedded in the Java source file to setup and compile the tool

You can modify the source code to change the output to your liking.

The tool uses XSLT style sheets to transform the event data into formatted text. You do not need to understand these style sheets. The tool will automatically load the correct style sheet, based on the event monitor type, and transform the event data. Each event monitor will provide default style sheets within the `sqllib/samples/xml/data` directory. The tool will also provide the following filtering options:

- Event ID
- Event timestamp
- Event type
- Workload name
- Service class name
- Application name

### Tool syntax

```
▶▶ java db2evmonfmt [connect XML file] [filter options] [ -h ]
```

#### connect:

```
| --d db_name --ue table_name [ -u user_id --p password ]
```

#### XML file:

```
| --f xml_filename
```

#### filter options:

```
| [ -fxml ] [ -ftext ] [ -ss stylesheet_name ] [ -id event_id ]
▶ [ -type event_type ] [ -hours num_hours ] [ -w workload_name ]
▶ [ -a appl_name ] [ -s srvc_subclass_name ]
```

### Tool parameters

#### java

To run the **db2evmonfmt** Java-based tool successfully, the **java** keyword must precede the tool name. The proper Java version to successfully run this tool is installed from the `sqllib/java/jdk64` directory during the DB2 product installation.

- d** *db\_name*  
Specifies the database name to which a connection is made
- ue** *table\_name*  
Specifies the name of the unformatted event table
- u** *user\_id*  
Specifies the user ID
- p** *password*  
Specifies the password
- f** *xml\_filename*  
Specifies the name of the input XML file to format
- fxml**  
Produces a formatted XML document (pipe to stdout)
- ftext**  
Formats an XML document to a text document (pipe to stdout)
- ss** *stylesheet\_name*  
Specifies the XSLT style sheet to use to transform the XML document
- id** *event\_id*  
Displays all events matching the specified event ID
- type** *event\_type*  
Displays all events matching the specified event type
- hours** *num\_hours*  
Displays all events that have occurred within the specified last number of hours
- w** *workload\_name*  
Displays all events that are part of the specified workload
- a** *appl\_name*  
Displays all events that are part of the specified application
- s** *svc\_subclass\_name*  
Displays all events that are part of the specified service subclass

### XSLT style sheets

The DB2 database manager provides default XSLT style sheets (see Table 1) which can be found in the `sql11ib/samples/java/jdbc` directory. You can change these style sheets to produce the required output.

*Table 17. Default XSLT style sheets for event monitors*

| Event monitor | Default XSLT style sheet |
|---------------|--------------------------|
| Locking       | DB2EvmonLocking.xml      |
| Unit of work  | DB2EvmonUOW.xml          |
| Package cache | DB2EvmonPkgCache.xml     |

You can create your own XSLT style sheet to transform XML documents. You can pass these style sheets into the Java-based tool using the `-ss stylesheet_name` option.

## Examples

### Example 1

To obtain a formatted text output for all events that have occurred in the last 32 hours from the package cache unformatted event table PKG in database SAMPLE, issue the following command:

```
java db2evmonfmt -d sample -ue pkg -ftext -hours 32
```

### Example 2

To obtain a formatted text output for all events of type LOCKTIMEOUT that have occurred in the last 24 hours from unformatted event table LOCK in database SAMPLE, issue the following command:

```
java db2evmonfmt -d sample -ue LOCK -ftext -hours 24 -type locktimeout
```

### Example 3

To obtain a formatted text output from the XML source file LOCK.XML, extracting all events that match the event type LOCKWAIT in the last 5 hours, issue the following command:

```
java db2evmonfmt -f lock.xml -ftext -type lockwait -hours 5
```

### Example 4

To obtain a formatted text output using the created XSLT style sheet SUMMARY.XSL for all events in the unformatted event table UOW in database SAMPLE, issue the following command:

```
java db2evmonfmt -d sample -ue uow -ftext -ss summary.xsl
```

## Sample formatted flat-text output

The following sample of formatted flat-text output was generated from the locking event monitor XSLT style sheet:

```
-----  
Event Entry      : 0  
Event ID        : 1  
Event Type      : Locktimeout  
Event Timestamp : 2008-05-23-12.00.14.132329000  
-----  
  
Lock Details  
-----  
Lock Name       : 020004010000000000000000054  
Lock Type      : Table  
Lock Attributes : 00000000  
Lock Count     : 1  
Lock Hold Count : 0  
Lock rrIID     : 0  
Lock Status    : Waiting  
Cursor Bitmap  : 00000000  
Tablespace Name : USERSPACE1  
Table Name     : NEWTON .SARAH  
  
Attributes      Requestor      Holder  
-----  
Application Handle [0-35]      [0-16]  
Application ID    *LOCAL.horton.080523160016 *LOCAL.horton.080523155938  
Application Name  xaplus0001  db2bp  
Authentication ID NEWTON      HORTON  
Requesting Agent  65         21  
Coordinating Agent 65         21  
Application Status SQLM_CONNECTPEND SQLM_CONNECTPEND  
Lock Timeout     5000      0  
Workload Name    XAPLUS0010_WL02 SYSDEFAULTUSERWORKLOAD  
Service Subclass XAPLUS0010_SC02 SYSDEFAULTSUBCLASS
```

```
Current Request      Execute
Lock Mode           Intent Exclusive
tpmon Userid
tpmon Wkstn
tpmon App
tpmon Accstring
```

```
Execute Immediate
Exclusive
```

Lock Requestor Current Activities

```
-----
Activity ID       : 2
Uow ID           : 1
Package ID       : 65426E4D4B584659
Package SectNo   : 3
Package Name     : NEWTON
Package Schema   : AKINTERF
Package Version  :
Reopt           : always
Eff Isolation    : Cursor Stability
Eff Locktimeout  : 5
Eff Degree       : 0
Nesting Level   : 0
Stmt Unicode     : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO SARAH VALUES(:H00008, :H00013, :H00014)
```

Lock Requestor Past Activities

```
-----
Activity ID       : 1
Uow ID           : 1
Package ID       : 65426E4D4B584659
Package SectNo   : 2
Package Name     : NEWTON
Package Schema   : AKINTERF
Package Version  :
Reopt           : always
Eff Isolation    : Cursor Stability
Eff Locktimeout  : 5
Eff Degree       : 0
Nesting Level   : 0
Stmt Unicode     : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO NADIA VALUES(:H00007)
```

Lock Holder Current Activities

Lock Holder Past Activities

```
-----
Activity ID       : 1
Uow ID           : 2
Package ID       : 41414141414E4758
Package SectNo   : 201
Package Name     : NULLID
Package Schema   : SQLC2G13
Package Version  :
Reopt           : none
Eff Isolation    : Cursor Stability
Eff Locktimeout  : 5
Eff Degree       : 0
Nesting Level   : 0
Stmt Unicode     : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Select (blockable)
```

```

Stmt Text      : select * from newton.sarah

Activity ID    : 2
Uow ID        : 2
Package ID     : 41414141414E4758
Package SectNo : 203
Package Name   : NULLID
Package Schema : SQLC2G13
Package Version :
Reopt         : none
Eff Isolation  : Cursor Stability
Eff Locktimeout : 5
Eff Degree     : 0
Nesting Level  : 0
Stmt Unicode   : No
Stmt Flag     : Dynamic
Stmt Type     : DML, Lock Table
Stmt Text     : lock table newton.sarah in exclusive mode

```

```

-----
Event Entry    : 1
Event ID      : 2
Event Type    : Locktimeout
Event Timestamp : 2008-05-23-12.04.42.144896000
-----

```

```

...
...
...

```

### Usage notes

The **db2evmonfmt** utility is a Java-based tool which must be preceded by the **java** keyword in order to run successfully. The Java version required is that which is installed with the DB2 product from the `sqllib/java/jdk64` directory.

**Note:** You can also use the `EVMON_FORMAT_UE_TO_XML` table function to format the binary events, contained in the unformatted event table BLOB column, into an XML document.

### Routines for extracting data from unformatted event tables:

If you want to perform queries on the data collected by an event monitor that writes to a unformatted event (UE) table, you must first extract the data from UE table using one of the two routines provided for this purpose. The `EVMON_FORMAT_UE_TO_TABLES` procedure extracts data from the UE table to create relational tables. The `EVMON_FORMAT_UE_TO_XML` table function creates an XML document.

#### **EVMON\_FORMAT\_UE\_TO\_TABLES**

The `EVMON_FORMAT_UE_TO_TABLES` procedure examines the UE table produced by an event monitor, and extracts the data it contains into relational tables that you can query. The number of tables produced depends on the type of event monitor; and the logical data groups for which that event monitor collects data. Generally speaking, the data from each logical data group is written to a separate table. For example, the package cache event monitor collects event data from three logical data groups: `pkgcache` and `pkgcache_metrics`, and `pkgcache_stmt_args`. Thus, three tables are produced by `EVMON_FORMAT_UE_TO_TABLES`.

**Note:** `EVMON_FORMAT_UE_TO_TABLES` does not create a table for the control logical data group.

In addition to creating relational tables from UE tables, as of Version 10.1 the `EVMON_FORMAT_UE_TO_TABLES` procedure provides the capability to prune data from UE tables. When you use the `PRUNE_UE_TABLES` option for `EVMON_FORMAT_UE_TO_TABLES`, data that is successfully inserted into relational tables is deleted from the unformatted event (UE) table.

### **EVMON\_FORMAT\_UE\_TO\_XML**

The `EVMON_FORMAT_UE_TO_XML` table function examines the UE table produced by an event monitor, and extracts the data it contains into an XML document. This document can then be queried as often as needed using `pureXML`.

#### **Notes:**

- This table function works similarly to the `db2evmonfmt` utility when that utility is used with the `-fxml` option. The differences between using `EVMON_FORMAT_UE_TO_XML` instead of `db2evmonfmt` are as follows:
  - `EVMON_FORMAT_UE_TO_XML` is a table function. As such, it is invoked as part of an SQL statement. `db2evmonfmt` runs as a separate utility.
  - `EVMON_FORMAT_UE_TO_XML` lets you specify a `SELECT` statement with a `WHERE` clause to filter events from the UE table. `db2evmonfmt` has only limited capabilities for filtering event data.
- The output XML document from `EVMON_FORMAT_UE_TO_XML` can be formatted by `db2evmonfmt` to create a flat text file.

With both routines, you must include a `SELECT` statement in the call to the routine to specify conditions for which data to extract.

#### **Pruning data from UE tables:**

If you use the `EVMON_FORMAT_UE_TO_TABLES` procedure to extract data from UE tables, you can use the `PRUNE_UE_TABLE` option to remove data that you no longer need.

#### **Before you begin**

Before you can extract data from a UE table, you must have created, activated, and enabled data collection for an event monitor that writes to a UE table.

#### **About this task**

In addition to the performance advantages that UE tables offer, using UE tables as output for an event monitor lets you take advantage of the automatic pruning feature of the `EVMON_FORMAT_UE_TO_TABLES` procedure. When you use this procedure, any data that is extracted from the UE table and written to a regular table can be automatically removed from the UE table. This procedure makes it easier to manage a UE table. For example, assume that you want to use a unit of work event monitor to capture information to generate daily reports for accounting purposes, such as charging departments for CPU time that is used by an application or query. In that case, you might want to prune that data after producing the reports.

## Procedure

To extract and then prune data from a UE table:

Issue an SQL statement that calls the `EVMON_FORMAT_UE_TO_TABLES` procedure with the `PRUNE_UE_TABLE` option to extract data into a regular table. For example, if you have a unit of work event monitor called `TRACKWORK`, you might create a statement such as the one that follows:

```
CALL EVMON_FORMAT_UE_TO_TABLES
 ('UOW', NULL, NULL, NULL, NULL, NULL, 'PRUNE_UE_TABLE', -1,
 'SELECT * FROM TRACKWORK')
```

All event data is copied from the UE table to the `UOW_EVENT_TRACKWORK` and `UOW_METRICS_TRACKWORK` tables. In addition, all records that were copied are removed from the UE table.

## Formatting file or pipe event monitor output from a command line

The output of a file or pipe event monitor is a binary stream of logical data groupings. You can format this data stream from a command line by using the `db2evmon` command. This productivity tool reads in event records from an event monitor's files or pipe, then writes them to the screen (standard output).

### Before you begin

No authorization is required unless you are connecting to the database, in which case one of the following authorities is required:

- SYSADM
- SYCTRL
- SYMAINT
- DBADM

### About this task

You can indicate which event monitor output to format by either providing the path of the event files, or providing the name of the database and the event monitor name.

## Procedure

To format event monitor output:

- Specify the directory containing the event monitor files:  
`db2evmon -path '/tmp/d1events'`

`/tmp/d1events` represents a (UNIX) path.

- Specify the database and event monitor name:  
`db2evmon -db 'sample' -evm 'd1mon'`

`sample` represents the database the event monitor belongs to.

`d1mon` represents an event monitor.

## Altering an event monitor

You cannot change an event monitor, with one exception: you can add one or more logical data groups to the set of logical data groups that the event monitor collects. You use the ALTER EVENT MONITOR statement to add logical groups.

### About this task

When you create an event monitor that writes to tables, by default, all logical data groups of monitor elements that are associated with that event monitor are captured. However, if you include the names of logical data groups in the CREATE EVENT MONITOR statement, only those groups are captured. For example, you might create an activities event monitor that captures data only from the event\_activity and event\_activity\_metrics logical data groups, as shown in the following example:

```
CREATE EVENT MONITOR myacts FOR ACTIVITIES
  WRITE TO TABLE
    event_activity, event_activity_metrics
```

The preceding DDL statement creates an event monitor that writes to two tables: ACTIVITY\_myacts and ACTIVITY\_METRICS\_myacts.

### Restrictions

You can use the ALTER EVENT MONITOR statement only to add logical data groups to an event monitor. You cannot remove a logical data group. You also cannot change the name, the target table space, or the value for PCTDEACTIVATE that is associated with the table that is used to capture the data in monitor elements that belong to a data group.

### Procedure

To add additional logical data groups to an event monitor:

1. Decide which logical data group you want to add. Using the preceding example of a locking event monitor where only two logical data groups are being captured, assume that you want to add the event\_activitystmt and event\_activityvals logical data groups.
2. Formulate an ALTER EVENT MONITOR statement to add these new logical data groups.

```
ALTER EVENT MONITOR mylacts
  ADD LOGICAL GROUP event_activitystmt
  ADD LOGICAL GROUP event_activityvals
```

3. Execute the statement.

### Results

When the ALTER EVENT MONITOR statement completes execution, two additional tables are created for the event monitor myacts:

```
ACTIVITYSTMT_myacts
ACTIVITYVALS_myacts
```

The next time the event monitor is activated, these tables are populated with data from their corresponding logical data groups.

**Remember:** If you add new logical data groups to an event monitor, any data that existed for the logical data groups that were originally part of the table will not

have any corresponding rows in the tables for the newly added logical group. Adjust your queries as needed, or consider pruning old data from the table after adding the logical groups.

## Example

A database administrator creates a locking event monitor called `mylocks` by using the following SQL statement:

```
CREATE EVENT MONITOR mylocks FOR LOCKING WRITE TO TABLE LOCK, LOCK_PARTICIPANTS
```

This statement collects information for monitor elements in the `lock` and `lock_participants` logical data groups. The tables to which the monitor element data is written are created with the default table names `LOCK_MYLOCKS` and `LOCK_PARTICIPANTS_MYLOCKS`.

Later on, the database administrator decides that she wants to collect information in the `LOCK_PARTICIPANT_ACTIVITIES` logical data group. She uses the following statement to modify the event monitor:

```
ALTER EVENT MONITOR mylocks ADD LOGICAL GROUP LOCK_PARTICIPANT_ACTIVITIES
```

This statement causes the monitor elements in the `lock_participant_activities` to be collected in addition to the other elements that already were collected. This new set of monitor elements are written to the table `LOCK_PARTICIPANT_ACTIVITIES_MYLOCKS`.

Later, the database administrator decides that she also needs the data from the `control` logical data group. However, she wants this data to be written to a table with a name other than the default name, and to a table space other than the default table space. She uses the following statement:

```
ALTER EVENT MONITOR mylocks ADD LOGICAL GROUP CONTROL TABLE ctl_mylocks IN mytbsp3
```

This statement adds the `control` logical data group to the output of the event monitor. This statement adds the `control` logical data group to the output of the event monitor. The data is written to the `CTL_MYLOCKS` table, and the table is written to the table space `mytbsp3`, instead of the default table space.

---

## Monitoring different types of events

### Lock and deadlock event monitoring

Diagnosing and correcting lock contention situations in large DB2 environments can be complex and time-consuming. The locking event monitor designed to simplify this task by collecting locking data.

**Note:** The `deadlocks` event monitor has been deprecated, and the function it provided is included in the locking event monitor. In addition, the `DB2DETAILDEADLOCK` event monitor is also deprecated. See “Deprecated lock monitoring functionality” on page 132 for important usage information about this event monitor.

The locking event monitor is used to capture descriptive information about lock events at the time that they occur. The information captured identifies the key applications involved in the lock contention that resulted in the lock event. Information is captured for both the lock requester (the application that received

the deadlock or lock timeout error, or waited for a lock for more than the specified amount of time) and the current lock owner.

The information collected by the lock event monitor can be written in binary format to an unformatted event table in the database, in which case the captured data must be processed in a post-capture step. Alternatively, the lock event information can be written to a set of regular tables. See “Output options for event monitors” on page 34 for more information about how to choose the most appropriate output format.

You can also directly access DB2 relational monitoring interfaces (table functions) to collect lock event information by using either dynamic or static SQL.

Determining if a deadlock or lock timeout has occurred is also simplified. Messages are written to the administration notification log when either of these events occurs; this supplements the SQL0911N (sqlcode -911) error returned to the application. In addition, a notification of lock escalations is also written to the administration notification log; this information can be useful in adjusting the size of the lock table and the amount of the table an application can use. There are also counters for lock timeouts (**lock\_timeouts**), lock waits (**lock\_waits**), and deadlocks (**deadlocks**) that can be checked.

The types of activities for which locking data can be captured are as follows:

- SQL statements, such as:
  - DML
  - DDL
  - CALL
- **LOAD** command
- **REORG** command
- **BACKUP DATABASE** command
- Utility requests

The lock event monitor replaces the deprecated deadlock event monitors (CREATE EVENT MONITOR FOR DEADLOCKS statement and DB2DETAILDEADLOCK) and the deprecated lock timeout reporting feature (DB2\_CAPTURE\_LOCKTIMEOUT registry variable) with a simplified and consistent interface for gathering locking event data, and adds the ability to capture data on lock waits.

## Functional overview

Two steps are required to enable the capturing of lock event data using the locking event monitor:

1. You must create a LOCK EVENT monitor using the CREATE EVENT MONITOR FOR LOCKING statement. You provide a name for the monitor and, if you are using UE tables as the output format, the name of an unformatted event table into which the lock event data is written.

**Note:** If you choose to use regular tables for event monitor output, default table names are assigned. You can override the defaults from the CREATE EVENT MONITOR statement, if you prefer.

2. You must specify the level for which you want lock event data captured by using one of the following methods:

- You can specify particular workloads by either altering an existing workload, or by creating a new workload using the CREATE or ALTER WORKLOAD statements. At the workload level you must specify the type of lock event data you want captured (deadlock, lock timeout or lock wait), and whether you want the SQL statement history and input values for the applications involved in the locking. For lock waits you must also specify the amount of time that an application will wait for a lock, after which data is captured for the lock wait.
- You can collect data at the database level and affect all DB2 workloads by setting the appropriate database configuration parameter:

**mon\_lockwait**

This parameter controls the generation of lock wait events

Best practice is to enable lock wait data collection at the workload level.

**mon\_locktimeout**

This parameter controls the generation of lock timeout events

Best practice is to enable lock timeout data collection at the database level if they are unexpected by the application. Otherwise enable at workload level.

**mon\_deadlock**

This parameter controls the generation of deadlock events

Best practice is to enable deadlock data collection at the database level.

**mon\_lw\_thresh**

This parameter controls the amount of time spent in lock wait before an event for **mon\_lockwait** is generated

Capturing of SQL statement history and input values uses additional processor time, memory and storage, but this level of detail is often needed to successfully debug a locking problem.

After a locking event has occurred, you can view the event data in the output produced by the event monitor. If you used UE tables, the binary data in the unformatted event table can be transformed into an XML or a text document using a supplied Java-based application called **db2evmonfmt**. In addition, you can format the binary event data in the unformatted event table BLOB column into either an XML report document, using the `EVMON_FORMAT_UE_TO_XML` table function, or into a relational table, using the `EVMON_FORMAT_UE_TO_TABLES` procedure.

If you used regular tables as the output format, you can query the data directly using SQL.

To aid in the determination of what workloads should be monitored for locking events, the administration notification log can be reviewed. Each time a deadlock or lock timeout is encountered, a message is written to the log. These messages identify the workload in which the lock requester and lock owner or owners are running, and the type of locking event. There are also counters at the workload level for lock timeouts (**lock\_timeouts**), lock waits (**lock\_waits**), and deadlocks (**deadlocks**) that can be checked.

## Information collected for a locking event

Some of the information for lock events collected by the lock event monitor include the following:

- The lock that resulted in an event
- The application holding the lock that resulted in the lock event
- The applications that were waiting for or requesting the lock that result in the lock event
- What the applications were doing during the lock event

## Deprecated lock monitoring functionality

The deprecated detailed deadlock event monitor, DB2DETAILDEADLOCK, is created by default for each database and starts when the database is activated. If you use the locking event monitor to detect deadlocks, consider disabling the DB2DETAILDEADLOCK event monitor. If the DB2DETAILDEADLOCK event monitor remains active while the locking event monitor also collects deadlock information, both event monitors will be collecting data, which can significantly affect performance.

To remove the DB2DETAILDEADLOCK event monitor, issue the following SQL statements:

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

## Data generated by locking event monitors

Locking event monitors produce data about locks and deadlocks in the system. You can choose to have the output from a locking event monitor to regular tables, or to an unformatted event (UE) table. If data is written to a UE table, you must perform post-processing on it to view the data.

Regardless of the output format you choose, locking event data comes from one of four logical groups:

- lock
- lock\_participants
- lock\_participant\_activities
- lock\_activity\_values

If you choose to have the locking event data written to regular tables, data from an additional group (CONTROL) is used to generate meta-data about the event monitor itself.

**Note:** By default, only deadlock information is generated for locking event monitors. To have other types of locking data generated, you must enable collection of that data explicitly.

## Information written to tables for a locking event monitor:

Information written by the locking event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for the locking event monitor, by default, five tables are produced, each containing monitor elements from one or more logical data groups:

Table 18. Tables produced by locking write-to-table event monitors

| Default table name                             | Logical data groups reported  |
|--|---|
| LOCK_ <i>evmon-name</i>                        | lock  |
| LOCK_PARTICIPANTS_ <i>evmon-name</i>           | lock_participants   |
| LOCK_PARTICIPANT_ACTIVITIES_ <i>evmon-name</i> | lock_participant_activities   |
| LOCK_ACTIVITY_VALUES_ <i>evmon-name</i>        | lock_activity_values  |
| CONTROL_ <i>evmon-name</i>                     | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

**Important:** Even though all five tables are produced by default, you must still ensure that data collection is enabled for the kind of lock information that you want to gather. Otherwise, some of the columns contain null values.

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced for the CREATE EVENT MONITOR or ALTER EVENT MONITOR statement. Refer to the reference topics for those statements for details.

### Tables produced

Table 19. Information returned for a locking event monitor: Default table name: LOCK\_ *evmon-name*

| Column name                | Data type             | Description  |
|----------------------------|-----------------------|--|
| PARTITION_KEY              | INTEGER               | “partition_key - Partitioning key monitor element” on page 1016  |
| DEADLOCK_TYPE              | VARCHAR(10)           | “deadlock_type - Deadlock type monitor element” on page 783      |
| DL_CONNS                   | INTEGER               | dl_conns - Connections involved in deadlock                      |
| EVENT_ID                   | BIGINT NOT NULL       | event_id - Event ID monitor element                              |
| EVENT_TIMESTAMP            | TIMESTAMP NOT NULL    | event_timestamp - Event timestamp monitor element                |
| EVENT_TYPE                 | VARCHAR(128) NOT NULL | event_type - Event Type monitor element monitor element          |
| MEMBER                     | SMALLINT NOT NULL     | member - Database member   |
| ROLLED_BACK_PARTICIPANT_NO | INTEGER               | rolled_back_participant_no - Rolled back application participant |

Table 20. Information returned for a locking event monitor: Default table name: LOCK\_PARTICIPANTS\_ *evmon-name*

| Column name   | Data type | Description   |
|---------------|-----------|---|
| PARTITION_KEY | INTEGER   | “partition_key - Partitioning key monitor element” on page 1016 |
| AGENT_STATUS  | INTEGER   | agent_status - DCS application agents                           |
| AGENT_TID     | BIGINT    | “agent_tid - Agent thread ID monitor element” on page 653       |

Table 20. Information returned for a locking event monitor: Default table name: LOCK\_PARTICIPANTS\_evmon-name (continued)

| Column name         | Data type    | Description   |
|---------------------|--------------|---|
| APPL_ACTION         | VARCHAR(64)  | "appl_action - Application action monitor element" on page 665                                |
| APPL_ID             | VARCHAR(128) | appl_id - Application ID  |
| APPL_NAME           | VARCHAR(128) | appl_name - Application name  |
| APPLICATION_HANDLE  | BIGINT       | application_handle - Application handle   |
| AUTH_ID             | VARCHAR(128) | auth_id - Authorization ID  |
| CLIENT_ACCTNG       | VARCHAR(255) | client_acctng - Client accounting string  |
| CLIENT_APPLNAME     | VARCHAR(255) | client_applname - Client application name   |
| CLIENT_USERID       | VARCHAR(255) | client_userid - Client user ID  |
| CLIENT_WRKSTNNAME   | VARCHAR(255) | client_wrkstnname - Client workstation name   |
| COORD_AGENT_TID     | BIGINT       | "coord_agent_tid - Coordinator agent engine dispatchable unit ID monitor element" on page 747 |
| CURRENT_REQUEST     | VARCHAR(32)  | "current_request - Current operation request monitor element" on page 766                     |
| DEADLOCK_MEMBER     | SMALLINT     | "deadlock_member - Deadlock member monitor element" on page 782                               |
| EVENT_ID            | BIGINT       | event_id - Event ID monitor element   |
| EVENT_TIMESTAMP     | TIMESTAMP    | event_timestamp - Event timestamp monitor element   |
| EVENT_TYPE          | VARCHAR(128) | event_type - Event Type monitor element monitor element                                       |
| INTERNAL_DATA       | VARCHAR(255) |   |
| LOCK_ATTRIBUTES     | CHAR(8)      | lock_attributes - Lock attributes   |
| LOCK_COUNT          | BIGINT       | lock_count - Lock count   |
| LOCK_CURRENT_MODE   | BIGINT       | lock_current_mode - Original lock mode before conversion                                      |
| LOCK_ESCALATION     | CHAR(3)      | lock_escalation - Lock escalation   |
| LOCK_HOLD_COUNT     | BIGINT       | lock_hold_count - Lock hold count   |
| LOCK_MODE           | BIGINT       | lock_mode - Lock mode   |
| LOCK_MODE_REQUESTED | BIGINT       | lock_mode_requested - Lock mode requested   |
| LOCK_NAME           | CHAR(32)     | lock_name - Lock name   |
| LOCK_OBJECT_TYPE    | BIGINT       | lock_object_type - Lock object type waited on   |
| LOCK_OBJECT_TYPE_ID | CHAR(1)      | Reserved for future use   |
| LOCK_RELEASE_FLAGS  | CHAR(8)      | lock_release_flags - Lock release flags   |
| LOCK_RRIID          | BIGINT       |   |
| LOCK_STATUS         | BIGINT       | lock_status - Lock status   |
| LOCK_TIMEOUT_VAL    | BIGINT       | lock_timeout_val - Lock timeout value   |

Table 20. Information returned for a locking event monitor: Default table name: LOCK\_PARTICIPANTS\_evmon-name (continued)

| Column name               | Data type                | Description  |
|---------------------------|--------------------------|--|
| LOCK_WAIT_END_TIME        | TIMESTAMP                | "lock_wait_end_time - Lock wait end timestamp monitor element" on page 923                   |
| LOCK_WAIT_START_TIME      | TIMESTAMP                | lock_wait_start_time - Lock wait start timestamp   |
| LOCK_WAIT_VAL             | BIGINT                   | "lock_wait_val - Lock wait value monitor element" on page 928                                |
| MEMBER                    | SMALLINT                 | member - Database member   |
| OBJECT_REQUESTED          | VARCHAR(10)              | "object_requested - Requested object monitor element" on page 992                            |
| PARTICIPANT_NO            | INTEGER                  | participant_no - Participant within deadlock   |
| PARTICIPANT_NO_HOLDING_LK | INTEGER                  | participant_no_holding_lk - Participant holding a lock on the object required by application |
| PARTICIPANT_TYPE          | VARCHAR(10)              | "participant_type - Participant type monitor element" on page 1016                           |
| PAST_ACTIVITIES_WRAPPED   | CHAR(3)                  | "past_activities_wrapped - Past activities list wrapped monitor element" on page 1018        |
| QUEUE_START_TIME          | TIMESTAMP                | "queue_start_time - Queue start timestamp monitor element" on page 1165                      |
| QUEUED_AGENTS             | BIGINT                   | "queued_agents - Queued threshold agents monitor element" on page 1166                       |
| SERVICE_CLASS_ID          | INTEGER                  | service_class_id - Service class ID  |
| SERVICE_SUBCLASS_NAME     | VARCHAR(128)             | service_subclass_name - Service subclass name  |
| SERVICE_SUPERCLASS_NAME   | VARCHAR(128)             | service_superclass_name - Service superclass name  |
| TABLE_FILE_ID             | BIGINT                   | table_file_id - Table file identification  |
| TABLE_NAME                | VARCHAR(128)             | table_name - Table name  |
| TABLE_SCHEMA              | VARCHAR(128)             | table_schema - Table schema name   |
| TABLESPACE_NAME           | VARCHAR(128)             | tablespace_name - Table space name   |
| THRESHOLD_ID              | INTEGER                  | "thresholdid - Threshold ID monitor element" on page 1308                                    |
| THRESHOLD_NAME            | VARCHAR(128)             | threshold_name - Threshold name  |
| UTILITY_INVOCATION_ID     | VARCHAR(32) FOR BIT DATA |  |
| WORKLOAD_ID               | INTEGER                  | workload_id - Workload ID  |
| WORKLOAD_NAME             | VARCHAR(128)             | workload_name - Workload name  |
| XID                       | VARCHAR(140)             | xid - Transaction ID   |

Table 21. Information returned for a locking event monitor: Default table name:  
 LOCK\_PARTICIPANT\_ACTIVITIES\_evmon-name

| Column name            | Data type    | Description   |
|------------------------|--------------|---|
| PARTITION_KEY          | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016               |
| ACTIVITY_ID            | BIGINT       | activity_id - Activity ID   |
| ACTIVITY_TYPE          | VARCHAR(10)  | activity_type - Activity type   |
| CONSISTENCY_TOKEN      | CHARACTER(8) | consistency_token - Package consistency token                                 |
| EFFECTIVE_ISOLATION    | CHARACTER(2) | effective_isolation - Effective isolation                                     |
| EFFECTIVE_QUERY_DEGREE | BIGINT       | effective_query_degree - Effective query degree                               |
| EVENT_ID               | BIGINT       | event_id - Event ID monitor element   |
| EVENT_TIMESTAMP        | TIMESTAMP    | event_timestamp - Event timestamp monitor element                             |
| EVENT_TYPE             | VARCHAR(128) | event_type - Event Type monitor element monitor element                       |
| INCREMENTAL_BIND       | CHARACTER(3) | "incremental_bind - Incremental bind monitor element" on page 874             |
| MEMBER                 | SMALLINT     | member - Database member  |
| PACKAGE_NAME           | VARCHAR(128) | package_name - Package name   |
| PACKAGE_SCHEMA         | VARCHAR(128) | package_schema - Package schema   |
| PACKAGE_VERSION_ID     | VARCHAR(64)  | package_version_id - Package version  |
| PARTICIPANT_NO         | SMALLINT     | participant_no - Participant within deadlock                                  |
| QUERY_ACTUAL_DEGREE    | INTEGER      | query_actual_degree - Actual runtime degree of intrapartition parallelism     |
| REOPT                  | VARCHAR(10)  | "reopt - Reopt bind option monitor element" on page 1176                      |
| SECTION_NUMBER         | BIGINT       | section_number - Section number   |
| STMT_FIRST_USE_TIME    | TIMESTAMP    | stmt_first_use_time - Statement first use timestamp                           |
| STMT_INVOCATION_ID     | BIGINT       | stmt_invocation_id - Statement invocation identifier                          |
| STMT_LAST_USE_TIME     | TIMESTAMP    | stmt_last_use_time - Statement last use timestamp                             |
| STMT_LOCK_TIMEOUT      | INTEGER      | stmt_lock_timeout - Statement lock timeout                                    |
| STMT_NEST_LEVEL        | BIGINT       | stmt_nest_level - Statement nesting level                                     |
| STMT_OPERATION         | VARCHAR(128) | "stmt_operation/operation - Statement operation monitor element" on page 1250 |
| STMT_PKG_CACHE_ID      | BIGINT       | stmt_pkgcache_id - Statement package cache identifier                         |
| STMT_QUERY_ID          | BIGINT       | stmt_query_id - Statement query identifier                                    |
| STMT_SOURCE_ID         | BIGINT       | stmt_source_id - Statement source identifier                                  |
| STMT_TEXT              | CLOB         | stmt_text - SQL statement text  |
| STMT_TYPE              | BIGINT       | stmt_type - Statement type  |
| STMT_UNICODE           | CHARACTER(3) | "stmt_unicode - Statement unicode flag monitor element" on page 1258          |
| UOW_ID                 | INTEGER      | uow_id - Unit of work ID  |

Table 22. Information returned for a locking event monitor: Default table name: LOCK\_ACTIVITY\_VALUES\_evmon-name

| Column name        | Data type     | Description   |
|--------------------|---------------|---|
| PARTITION_KEY      | INTEGER       | "partition_key - Partitioning key monitor element" on page 1016 |
| ACTIVITY_ID        | BIGINT        | activity_id - Activity ID                                       |
| EVENT_ID           | BIGINT        | event_id - Event ID monitor element                             |
| EVENT_TIMESTAMP    | TIMESTAMP     | event_timestamp - Event timestamp monitor element               |
| EVENT_TYPE         | VARCHAR(128)  | event_type - Event Type monitor element monitor element         |
| MEMBER             | SMALLINT      | member - Database member  |
| PARTICIPANT_NO     | SMALLINT      | participant_no - Participant within deadlock                    |
| STMT_VALUE_DATA    | CLOB          | stmt_value_data - Value data                                    |
| STMT_VALUE_INDEX   | INTEGER       | stmt_value_index - Value index                                  |
| STMT_VALUE_ISNULL  | INTEGER       | stmt_value_isnull - Value has null value                        |
| STMT_VALUE_ISREOPT | INTEGER       | stmt_value_isreopt - Variable used for statement reoptimization |
| STMT_VALUE_TYPE    | CHARACTER(16) | stmt_value_type - Value type                                    |
| UOW_ID             | INTEGER       | uow_id - Unit of work ID  |

Table 23. Information returned for a locking event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

**Information written to relational tables by EVMON\_FORMAT\_UE\_TO\_TABLES for a locking event monitor:**

Information written for a locking event monitor from the EVMON\_FORMAT\_UE\_TO\_TABLES table function. This is also documented in the sql1lib/misc/DB2EvmonLocking.xsd file.

Table 24. Information returned for a locking event monitor: Table name: LOCK\_EVENT

| Column Name   | Data Type             | Description   |
|---------------|-----------------------|---|
| XMLID         | VARCHAR(256) NOT NULL | "xmlid - XML ID monitor element" on page 1431               |
| DEADLOCK_TYPE | VARCHAR(10)           | "deadlock_type - Deadlock type monitor element" on page 783 |
| EVENT_ID      | BIGINT NOT NULL       | event_id - Event ID monitor element                         |

Table 24. Information returned for a locking event monitor: Table name: LOCK\_EVENT (continued)

| Column Name                | Data Type             | Description  |
|----------------------------|-----------------------|--|
| EVENT_TYPE                 | VARCHAR(128) NOT NULL | event_type - Event Type monitor element monitor element          |
| EVENT_TIMESTAMP            | TIMESTAMP NOT NULL    | event_timestamp - Event timestamp monitor element                |
| MEMBER                     | SMALLINT NOT NULL     | member - Database member   |
| DL_CONNS                   | INTEGER               | dl_conns - Connections involved in deadlock                      |
| ROLLED_BACK_PARTICIPANT_NO | INTEGER               | rolled_back_participant_no - Rolled back application participant |

Table 25. Information returned for a locking event monitor: Table name: LOCK\_PARTICIPANTS

| Column Name               | Data Type             | Description   |
|---------------------------|-----------------------|---|
| XMLID                     | VARCHAR(256) NOT NULL | "xmlid - XML ID monitor element" on page 1431   |
| PARTICIPANT_NO            | INTEGER               | participant_no - Participant within deadlock  |
| PARTICIPANT_TYPE          | VARCHAR(10)           | "participant_type - Participant type monitor element" on page 1016                            |
| PARTICIPANT_NO_HOLDING_LK | INTEGER               | participant_no_holding_lk - Participant holding a lock on the object required by application  |
| APPLICATION_HANDLE        | BIGINT                | application_handle - Application handle   |
| APPL_ACTION               | VARCHAR(64)           | "appl_action - Application action monitor element" on page 665                                |
| APPL_ID                   | VARCHAR(128)          | appl_id - Application ID  |
| APPL_NAME                 | VARCHAR(128)          | appl_name - Application name  |
| AUTH_ID                   | VARCHAR(128)          | auth_id - Authorization ID  |
| AGENT_TID                 | BIGINT                | "agent_tid - Agent thread ID monitor element" on page 653                                     |
| COORD_AGENT_TID           | BIGINT                | "coord_agent_tid - Coordinator agent engine dispatchable unit ID monitor element" on page 747 |
| AGENT_STATUS              | INTEGER               | agent_status - DCS application agents   |
| DEADLOCK_MEMBER           | SMALLINT              | "deadlock_member - Deadlock member monitor element" on page 782                               |
| LOCK_TIMEOUT_VAL          | BIGINT                | lock_timeout_val - Lock timeout value   |

Table 25. Information returned for a locking event monitor: Table name: LOCK\_PARTICIPANTS (continued)

| Column Name             | Data Type            | Description   |
|-------------------------|----------------------|---|
| LOCK_WAIT_VAL           | BIGINT               | "lock_wait_val - Lock wait value monitor element" on page 928                         |
| WORKLOAD_ID             | INTEGER              | workload_id - Workload ID   |
| WORKLOAD_NAME           | VARCHAR(128)         | workload_name - Workload name   |
| SERVICE_CLASS_ID        | INTEGER              | service_class_id - Service class ID   |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128)         | service_superclass_name - Service superclass name                                     |
| SERVICE_SUBCLASS_NAME   | VARCHAR(128)         | service_subclass_name - Service subclass name   |
| CURRENT_REQUEST         | VARCHAR(32)          | "current_request - Current operation request monitor element" on page 766             |
| LOCK_ESCALATION         | CHAR(3)              | lock_escalation - Lock escalation   |
| PAST_ACTIVITIES_WRAPPED | CHAR(3)              | "past_activities_wrapped - Past activities list wrapped monitor element" on page 1018 |
| CLIENT_USERID           | VARCHAR(255)         | client_userid - Client user ID  |
| CLIENT_WRKSTNNAME       | VARCHAR(255)         | client_wrkstnname - Client workstation name   |
| CLIENT_APPLNAME         | VARCHAR(255)         | client_applname - Client application name   |
| CLIENT_ACCTNG           | VARCHAR(255)         | client_acctng - Client accounting string  |
| OBJECT_REQUESTED        | VARCHAR(10)          | "object_requested - Requested object monitor element" on page 992                     |
| LOCK_NAME               | CHAR(32)             | lock_name - Lock name   |
| LOCK_OBJECT_TYPE        | VARCHAR(32)          | lock_object_type - Lock object type waited on   |
| LOCK_OBJECT_TYPE_ID     | CHAR(1) FOR BIT DATA | Reserved for future use   |
| LOCK_ATTRIBUTES         | CHAR(8)              | lock_attributes - Lock attributes   |
| LOCK_CURRENT_MODE       | BIGINT               | lock_current_mode - Original lock mode before conversion                              |
| LOCK_MODE_REQUESTED     | BIGINT               | lock_mode_requested - Lock mode requested   |
| LOCK_MODE               | BIGINT               | lock_mode - Lock mode   |
| LOCK_COUNT              | BIGINT               | lock_count - Lock count   |
| LOCK_HOLD_COUNT         | BIGINT               | lock_hold_count - Lock hold count   |
| LOCK_RRIID              | BIGINT               |   |
| LOCK_STATUS             | BIGINT               | lock_status - Lock status   |
| LOCK_RELEASE_FLAGS      | CHAR(8)              | lock_release_flags - Lock release flags   |

Table 25. Information returned for a locking event monitor: Table name: LOCK\_PARTICIPANTS (continued)

| Column Name           | Data Type                  | Description  |
|-----------------------|----------------------------|--|
| LOCK_WAIT_START_TIME  | TIMESTAMP                  | lock_wait_start_time - Lock wait start timestamp                           |
| LOCK_WAIT_END_TIME    | TIMESTAMP                  | “lock_wait_end_time - Lock wait end timestamp monitor element” on page 923 |
| QUEUED_AGENTS         | BIGINT                     | “queued_agents - Queued threshold agents monitor element” on page 1166     |
| QUEUE_START_TIME      | TIMESTAMP                  | “queue_start_time - Queue start timestamp monitor element” on page 1165    |
| TABLE_FILE_ID         | BIGINT                     | table_file_id - Table file identification                                  |
| TABLE_NAME            | VARCHAR(128)               | table_name - Table name  |
| TABLE_SCHEMA          | VARCHAR(128)               | table_schema - Table schema name   |
| TABLESPACE_NAME       | VARCHAR(128)               | tablespace_name - Table space name   |
| THRESHOLD_ID          | INTEGER                    | “thresholdid - Threshold ID monitor element” on page 1308                  |
| THRESHOLD_NAME        | VARCHAR(128)               | threshold_name - Threshold name  |
| UTILITY_INVOCATION_ID | VARCHAR(32) FOR BIT DATA   |  |
| XID                   | VARCHAR(140) FOR BIT DATA) | xid - Transaction ID   |

Table 26. Information returned for a locking event monitor: Table name: LOCK\_PARTICIPANT\_ACTIVITIES

| Column Name        | Data Type             | Description  |
|--------------------|-----------------------|--|
| XMLID              | VARCHAR(256) NOT NULL | “xmlid - XML ID monitor element” on page 1431            |
| PARTICIPANT_NO     | INTEGER               | participant_no - Participant within deadlock             |
| ACTIVITY_ID        | INTEGER               | activity_id - Activity ID                                |
| ACTIVITY_TYPE      | VARCHAR(10)           | activity_type - Activity type                            |
| UOW_ID             | INTEGER               | uow_id - Unit of work ID                                 |
| PACKAGE_NAME       | VARCHAR(128)          | package_name - Package name                              |
| PACKAGE_SCHEMA     | VARCHAR(128)          | package_schema - Package schema                          |
| PACKAGE_VERSION_ID | VARCHAR(64)           | package_version_id - Package version                     |
| CONSISTENCY_TOKEN  | VARCHAR(8)            | consistency_token - Package consistency token            |
| SECTION_NUMBER     | BIGINT                | section_number - Section number                          |
| REOPT              | VARCHAR(10)           | “reopt - Reopt bind option monitor element” on page 1176 |

Table 26. Information returned for a locking event monitor: Table name: LOCK\_PARTICIPANT\_ACTIVITIES (continued)

| Column Name            | Data Type     | Description   |
|------------------------|---------------|---|
| INCREMENTAL_BIND       | CHAR(3)       | “incremental_bind - Incremental bind monitor element” on page 874             |
| EFFECTIVE_ISOLATION    | CHAR(2)       | effective_isolation - Effective isolation                                     |
| EFFECTIVE_QUERY_DEGREE | BIGINT        | effective_query_degree - Effective query degree                               |
| STMT_LOCK_TIMEOUT      | INTEGER       | stmt_lock_timeout - Statement lock timeout                                    |
| STMT_TYPE              | BIGINT        | stmt_type - Statement type  |
| STMT_QUERY_ID          | BIGINT        | stmt_query_id - Statement query identifier                                    |
| STMT_NEST_LEVEL        | BIGINT        | stmt_nest_level - Statement nesting level                                     |
| STMT_INVOCATION_ID     | BIGINT        | stmt_invocation_id - Statement invocation identifier                          |
| STMT_OPERATION         | VARCHAR(128)  | “stmt_operation/operation - Statement operation monitor element” on page 1250 |
| STMT_SOURCE_ID         | BIGINT        | stmt_source_id - Statement source identifier                                  |
| STMT_PKG_CACHE_ID      | BIGINT        | stmt_pkgcache_id - Statement package cache identifier                         |
| STMT_FIRST_USE_TIME    | TIMESTAMP     | stmt_first_use_time - Statement first use timestamp                           |
| STMT_LAST_USE_TIME     | TIMESTAMP     | stmt_last_use_time - Statement last use timestamp                             |
| STMT_TEXT              | CLOB(2097152) | stmt_text - SQL statement text  |
| STMT_UNICODE           | CHAR(3)       | “stmt_unicode - Statement unicode flag monitor element” on page 1258          |
| QUERY_ACTUAL_DEGREE    | INTEGER       | query_actual_degree - Actual runtime degree of intrapartition parallelism     |

Table 27. Information returned for a locking event monitor: Table name: LOCK\_ACTIVITY\_VALUES

| Column Name      | Data Type             | Description                                   |
|------------------|-----------------------|---|
| XMLID            | VARCHAR(256) NOT NULL | “xmlid - XML ID monitor element” on page 1431 |
| PARTICIPANT_NO   | INTEGER               | participant_no - Participant within deadlock  |
| ACTIVITY_ID      | INTEGER               | activity_id - Activity ID                     |
| UOW_ID           | INTEGER               | uow_id - Unit of work ID                      |
| STMT_VALUE_INDEX | INTEGER               | stmt_value_index - Value index                |

Table 27. Information returned for a locking event monitor: Table name: LOCK\_ACTIVITY\_VALUES (continued)

| Column Name        | Data Type | Description   |
|--------------------|-----------|---|
| STMT_VALUE_ISREOPT | INTEGER   | stmt_value_isreopt - Variable used for statement reoptimization |
| STMT_VALUE_ISNULL  | INTEGER   | stmt_value_isnull - Value has null value                        |
| STMT_VALUE_TYPE    | CHAR(16)  | stmt_value_type - Value type                                    |
| STMT_VALUE_DATA    | CLOB(32K) | stmt_value_data - Value data                                    |

**Information written to XML by EVMON\_FORMAT\_UE\_TO\_XML for a locking event monitor:**

Information written for a locking event monitor from the EVMON\_FORMAT\_UE\_TO\_XML table function. This is also documented in the sql1lib/misc/DB2EvmonLocking.xsd file.

**db2\_lock\_event**

The main schema that describes a lock timeout, lock wait or deadlock event in details.

**Element content:** ( ( "db2\_deadlock\_graph" {zero or one times (?)} , "db2\_participant" on page 143 {one or more (+)} ) | ( "db2\_message" on page 143, "db2\_event\_file" on page 143 ) )

**Attributes:**

| QName                            | Type        | Fixed | Default | Use      | Annotation |
|----------------------------------|-------------|-------|---------|----------|------------|
| id                               | xs:long     |       |         | required |            |
| type                             |             |       |         | required |            |
| timestamp                        | xs:dateTime |       |         | required |            |
| member                           |             |       |         | required |            |
| release                          | xs:long     |       |         | required |            |
| ANY attribute from ANY namespace |             |       |         |          |            |

**db2\_deadlock\_graph**

Schema element represents the DB2 Deadlock Graph. The graph outlines all the participants involved in the deadlock.

**Contained by:** "db2\_lock\_event"

**Element content:** ( "db2\_participant" on page 153 {one or more (+)} )

**Attributes:**

| QName    | Type   | Fixed | Default | Use      | Annotation |
|----------|--------|-------|---------|----------|------------|
| dl_conns | xs:int |       |         | required |            |

| QName                            | Type   | Fixed | Default  | Use      | Annotation |
|----------------------------------|--------|-------|----------|----------|------------|
| rolled_back_participant_no       | xs:int |       |          | required |            |
| type                             |        |       | required |          |            |
| ANY attribute from ANY namespace |        |       |          |          |            |

### db2\_participant

Schema element represents the application information of the all the participants involved in a lock event.

**Contained by:** “db2\_lock\_event” on page 142 “db2\_deadlock\_graph” on page 142

**Element content:** ( “db2\_object\_requested” on page 148 {zero or one times (?)} , “db2\_app\_details” on page 148, “db2\_activity” on page 148 {zero or more (\*)} )

#### Attributes:

| QName                            | Type   | Fixed | Default  | Use      | Annotation |
|----------------------------------|--------|-------|----------|----------|------------|
| no                               | xs:int |       |          | required |            |
| type                             |        |       | required |          |            |
| participant_no_holding_lk        | xs:int |       |          | optional |            |
| deadlock_member                  | xs:int |       |          | optional |            |
| ANY attribute from ANY namespace |        |       |          |          |            |

### db2\_message

Error message

**Contained by:** “db2\_lock\_event” on page 142

### db2\_event\_file

Fully qualified path to file where event has been written.

**Contained by:** “db2\_lock\_event” on page 142

### application\_handle

A system-wide unique ID for the application. See monitor element “agent\_id - Application handle (agent ID) monitor element” on page 650for more details.

**Contained by:** “db2\_app\_details” on page 148

### appl\_id

This identifier is generated when the application connects to the database at the database manager. See monitor element “appl\_id - Application ID monitor element” on page 666for more details.

**Contained by:** "db2\_app\_details" on page 148

### **appl\_name**

The name of the application running at the client, as known to the database. See monitor element "appl\_name - Application name monitor element" on page 670 for more details.

**Contained by:** "db2\_app\_details" on page 148

### **auth\_id**

The authorization ID of the user who invoked the application that is being monitored. See monitor element "auth\_id - Authorization ID" on page 686 for more details.

**Contained by:** "db2\_app\_details" on page 148

### **agent\_tid**

**Contained by:** "db2\_app\_details" on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **coord\_agent\_tid**

**Contained by:** "db2\_app\_details" on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **agent\_status**

The current status of the application. See monitor element "appl\_status - Application status monitor element" on page 673 for more details.

**Contained by:** "db2\_app\_details" on page 148

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | optional |            |

### **appl\_action**

The action/request that the client application is performing

**Contained by:** "db2\_app\_details" on page 148

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | optional |            |

**lock\_timeout\_val**

The database configuration parameter lock timeout. Value in seconds. See monitor element “lock\_timeout\_val - Lock timeout value monitor element” on page 920 for more details.

**Contained by:** “db2\_app\_details” on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**lock\_wait\_val**

The lock wait parameter in effect during the lock event. This is either the database configuration parameter MON\_LKWAIT\_THRSH or the COLLECT LOCK WAIT DATA setting specified at the workload level. Value in milliseconds.

**Contained by:** “db2\_app\_details” on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**tentry\_state**

TEntry state. Internal use only.

**Contained by:** “db2\_app\_details” on page 148

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | optional |            |

**tentry\_flag1**

TEntry flags1. Internal use only.

**Contained by:** “db2\_app\_details” on page 148

**tentry\_flag2**

TEntry flags2. Internal use only.

**Contained by:** "db2\_app\_details" on page 148

**xid**

XID - Global transaction identifier

**Contained by:** "db2\_app\_details" on page 148

**workload\_id**

ID of the workload to which this application belongs. See monitor element "workload\_id - Workload ID monitor element" on page 1426 for more details.

**Contained by:** "db2\_app\_details" on page 148

**workload\_name**

Name of the workload to which this application belongs. See monitor element "workload\_name - Workload name monitor element" on page 1427 for more details.

**Contained by:** "db2\_app\_details" on page 148

**service\_class\_id**

ID of the service subclass to which this application belongs. See monitor element "service\_class\_id - Service class ID monitor element" on page 1207 for more details.

**Contained by:** "db2\_app\_details" on page 148

**service\_subclass\_name**

Name of the service subclass to which this application belongs. See monitor element "service\_subclass\_name - Service subclass name monitor element" on page 1208 for more details.

**Contained by:** "db2\_app\_details" on page 148

**current\_request**

The operation currently being processed or most recently processed.

**Contained by:** "db2\_app\_details" on page 148

**lock\_escalation**

Indicates whether a lock request was made as part of a lock escalation. See monitor element "lock\_escalation - Lock escalation monitor element" on page 904 for more details. Possible values: Yes or No.

**Contained by:** "db2\_app\_details" on page 148

### **past\_activities\_wrapped**

Indicates whether the activities list has wrapped. The default limit on the number of past activities to be kept by any one application is 250. This default can be overridden using the registry variable DB2\_MAX\_INACT\_STMTS. Users may want to choose a different value for the limit to increase or reduce the amount of system monitor heap used for inactive statement information.

**Contained by:** “db2\_app\_details” on page 148

### **client\_userid**

The client user ID generated by a transaction manager and provided to the server. See monitor element “client\_userid - Client user ID monitor element” on page 717 for more details.

**Contained by:** “db2\_app\_details” on page 148

### **client\_wrkstname**

Identifies the client system or workstation, if the sqleseti API was issued in this connection. See monitor element “client\_wrkstname - Client workstation name monitor element” on page 717 for more details.

**Contained by:** “db2\_app\_details” on page 148

### **client\_applname**

Identifies the server transaction program performing the transaction, if the sqleseti API was issued in this connection. See monitor element “client\_applname - Client application name monitor element” on page 710 for more details.

**Contained by:** “db2\_app\_details” on page 148

### **client\_acctng**

The data passed to the target database for logging and diagnostic purposes, if the sqleseti API was issued in this connection. See monitor element “client\_acctng - Client accounting string monitor element” on page 709 for more details.

**Contained by:** “db2\_app\_details” on page 148

### **utility\_invocation\_id**

**Contained by:** “db2\_app\_details” on page 148

### **service\_superclass\_name**

Name of the service superclass to which this application belongs. See monitor element “service\_superclass\_name - Service superclass name monitor element” on page 1209 for more details.

**Contained by:** “db2\_app\_details” on page 148

## db2\_object\_requested

Schema element represents the DB2 lock that the Requestor is attempting to acquire, which is being held by the Owner.

**Contained by:** “db2\_participant” on page 143

**Element content:** ( ( “lock\_name” on page 149, “lock\_object\_type” on page 149, “lock\_specifics” on page 149, “lock\_attributes” on page 149, “lock\_current\_mode” on page 149, “lock\_mode\_requested” on page 150, “lock\_mode” on page 150, “lock\_count” on page 150, “lock\_hold\_count” on page 150, “lock\_rriid” on page 151, “lock\_status” on page 151, “lock\_release\_flags” on page 151, “tablespace\_name” on page 151, “table\_name” on page 151, “table\_schema” on page 152, “lock\_object\_type\_id” on page 152, “lock\_wait\_start\_time” on page 152, “lock\_wait\_end\_time” on page 152, ANY content ( skip ) {zero or more (\*)} ) | ( “threshold\_name” on page 153, “threshold\_id” on page 153, “queued\_agents” on page 153, “queue\_start\_time” on page 153, ANY content ( skip ) {zero or more (\*)} ) )

**Attributes:**

| QName | Type | Fixed | Default  | Use | Annotation |
|-------|------|-------|----------|-----|------------|
| type  |      |       | required |     |            |

## db2\_app\_details

Schema element represents the details regarding this participant.

**Contained by:** “db2\_participant” on page 143

**Element content:** ( “application\_handle” on page 143, “appl\_id” on page 143, “appl\_name” on page 144, “auth\_id” on page 144, “agent\_tid” on page 144, “coord\_agent\_tid” on page 144, “agent\_status” on page 144, “appl\_action” on page 144, “lock\_timeout\_val” on page 145, “lock\_wait\_val” on page 145, “tentry\_state” on page 145, “tentry\_flag1” on page 145, “tentry\_flag2” on page 146, “xid” on page 146, “workload\_id” on page 146, “workload\_name” on page 146, “service\_class\_id” on page 146, “service\_subclass\_name” on page 146, “current\_request” on page 146, “lock\_escalation” on page 146, “past\_activities\_wrapped” on page 147, “client\_userid” on page 147, “client\_wrkstnname” on page 147, “client\_applname” on page 147, “client\_acctng” on page 147, “utility\_invocation\_id” on page 147, “service\_superclass\_name” on page 147, ANY content ( skip ) {zero or more (\*)} )

## db2\_activity

List of all DB2 activities the application is currently executing or has executed.

**Contained by:** “db2\_participant” on page 143

**Element content:** ( “db2\_activity\_details” on page 158, “db2\_input\_variable” on page 159 {zero or more (\*)} )

**Attributes:**

| QName | Type | Fixed | Default  | Use | Annotation |
|-------|------|-------|----------|-----|------------|
| type  |      |       | required |     |            |

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### lock\_name

Internal binary lock name. This element serves as a unique identifier for locks. See monitor element “lock\_name - Lock name monitor element” on page 914 for more details.

**Contained by:** “db2\_object\_requested” on page 148

### lock\_object\_type

The type of object the application is waiting to obtain a lock. See monitor element “lock\_object\_type - Lock object type waited on monitor element” on page 916 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |

### lock\_specifics

Internal specifics about the lock. For information use only.

**Contained by:** “db2\_object\_requested” on page 148

### lock\_attributes

Lock attributes. See monitor element “lock\_attributes - Lock attributes monitor element” on page 901 for more details.

**Contained by:** “db2\_object\_requested” on page 148

### lock\_current\_mode

Original lock before conversion. See monitor element “lock\_current\_mode - Original lock mode before conversion monitor element” on page 903 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |
| mode  |         |       |         | optional |            |

### lock\_mode\_requested

The lock mode being requested by this participant. See monitor element “lock\_mode\_requested - Lock mode requested monitor element” on page 913 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |
| mode  |         |       |         | optional |            |

### lock\_mode

The type of lock being held. See monitor element “lock\_mode - Lock mode monitor element” on page 912 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |
| mode  |         |       |         | optional |            |

### lock\_count

The number of locks on the lock being held. See monitor element “lock\_count - Lock count monitor element” on page 902 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_hold\_count

The number of holds placed on the lock. See monitor element “lock\_hold\_count - Lock hold count monitor element” on page 911 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_rriid

IID for Row locking. Internal use only.

**Contained by:** “db2\_object\_requested” on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_status

Indicates the internal status of the lock. See monitor element “lock\_status - Lock status monitor element” on page 919for more details.

**Contained by:** “db2\_object\_requested” on page 148

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | optional |            |

### lock\_release\_flags

Lock release flags. See monitor element “lock\_release\_flags - Lock release flags monitor element” on page 918for more details.

**Contained by:** “db2\_object\_requested” on page 148

### tablespace\_name

The name of the table space where the lock is held. See monitor element “tablespace\_name - Table space name monitor element” on page 1279for more details.

**Contained by:** “db2\_object\_requested” on page 148

**Attributes:**

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |

### table\_name

The name of the table where the lock is held. See monitor element “table\_name - Table name monitor element” on page 1269for more details.

**Contained by:** “db2\_object\_requested” on page 148

**Attributes:**

| QName          | Type    | Fixed | Default | Use      | Annotation   |
|----------------|---------|-------|---------|----------|--|
| id             | xs:long |       |         | optional |  |
| data_member_id |         |       |         | optional | The identifier of the data member for which information is returned. |

### **table\_schema**

The schema of the table. See monitor element “table\_schema - Table schema name monitor element” on page 1271 for more details.

**Contained by:** “db2\_object\_requested” on page 148

### **lock\_object\_type\_id**

The type of object the application is waiting to obtain a lock. See monitor element “lock\_object\_type - Lock object type waited on monitor element” on page 916 for more details.

**Contained by:** “db2\_object\_requested” on page 148

### **lock\_wait\_start\_time**

The data and time the application started waiting to obtain a lock on the object that is currently locked by the lock owner. See monitor element “lock\_wait\_start\_time - Lock wait start timestamp monitor element” on page 923 for more details.

**Contained by:** “db2\_object\_requested” on page 148

#### **Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### **lock\_wait\_end\_time**

The data and time the application stopped waiting to obtain a lock on the object that is currently locked by the lock owner.

**Contained by:** “db2\_object\_requested” on page 148

#### **Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

**threshold\_name**

The name of the threshold queue.

**Contained by:** “db2\_object\_requested” on page 148

**threshold\_id**

The ID of the threshold queue.

**Contained by:** “db2\_object\_requested” on page 148

**queued\_agents**

The total number of agents currently queued in the threshold.

**Contained by:** “db2\_object\_requested” on page 148

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**queue\_start\_time**

The data and time the application started waiting in the queue to to obtain a threshold ticket.

**Contained by:** “db2\_object\_requested” on page 148

**Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

**db2\_participant**

Schema element represents a single stack entry in a deadlock graph.

**Contained by:** “db2\_lock\_event” on page 142 “db2\_deadlock\_graph” on page 142

**Attributes:**

| QName                               | Type   | Fixed | Default | Use      | Annotation |
|-------------------------------------|--------|-------|---------|----------|------------|
| no                                  | xs:int |       |         | required |            |
| deadlock_member                     |        |       |         | required |            |
| participant_no<br>_holding_lk       | xs:int |       |         | required |            |
| application_handle                  |        |       |         | required |            |
| ANY attribute from<br>ANY namespace |        |       |         |          |            |

### **activity\_id**

Counter which uniquely identifies an activity for an application within a given unit of work. See monitor element “activity\_id - Activity ID monitor element” on page 645 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **uow\_id**

The unit of work ID to which this activity record applies. See monitor element “uow\_id - Unit of work ID monitor element” on page 1401 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **package\_name**

The name of the package that contains the SQL statement currently executing. See monitor element “package\_name - Package name monitor element” on page 1006 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **package\_schema**

The schema name of the package associated with an SQL statement. See monitor element “package\_schema - Package schema monitor element” on page 1007 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **package\_version\_id**

The package version identifies the version identifier of the package that contains the SQL statement currently executing. See monitor element “package\_version\_id - Package version monitor element” on page 1007 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **consistency\_token**

The package consistency token helps to identify the version of the package that contains the SQL statement currently executing. See monitor element “consistency\_token - Package consistency token monitor element” on page 738 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### **section\_number**

The internal section number in the package for the SQL statement currently processing or most recently processed. See monitor element “section\_number - Section number monitor element” on page 1201 for more details.

**Contained by:** “db2\_activity\_details” on page 158

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**reopt**

The REOPT bind option used to precompile this package. Possible values are: NONE, ONCE, and ALWAYS. See the REOPT bind options for more details.

**Contained by:** “db2\_activity\_details” on page 158

**incremental\_bind**

The package was incrementally bound at execution time. Possible values: Yes or No.

**Contained by:** “db2\_activity\_details” on page 158

**effective\_isolation**

The isolation value in effect for the SQL statement while it was being run. See monitor element “effective\_isolation - Effective isolation monitor element” on page 805 for more details.

**Contained by:** “db2\_activity\_details” on page 158

**Attributes:**

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |

**effective\_query\_degree**

The degree value in effect for the SQL statement while it was being run. See monitor element “effective\_query\_degree - Effective query degree monitor element” on page 806 for more details.

**Contained by:** “db2\_activity\_details” on page 158

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**stmt\_unicode**

The SQL statement unicode flag. Possible values: Yes or No.

**Contained by:** “db2\_activity\_details” on page 158

### stmt\_lock\_timeout

The locktimeout value in effect for the SQL statement while it was being run. See monitor element "stmt\_lock\_timeout - Statement lock timeout monitor element" on page 1249for more details

**Contained by:** "db2\_activity\_details" on page 158

#### Element content:

| Type   | Facet |
|--------|-------|
| xs:int |       |

### stmt\_type

The type of SQL statement processed. Possible values: Dynamic or Static. See monitor element "stmt\_type - Statement type monitor element" on page 1256for more details.

**Contained by:** "db2\_activity\_details" on page 158

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | required |            |

### stmt\_operation

**Contained by:** "db2\_activity\_details" on page 158

### stmt\_query\_id

Internal query identifier given to any SQL statement. See monitor element "stmt\_query\_id - Statement query identifier monitor element" on page 1252for more details.

**Contained by:** "db2\_activity\_details" on page 158

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### stmt\_nest\_level

This element shows the level of nesting or recursion in effect when the statement was run. See monitor element "stmt\_nest\_level - Statement nesting level monitor element" on page 1249for more details.

**Contained by:** "db2\_activity\_details" on page 158

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **stmt\_invocation\_id**

This element shows the identifier of the routine invocation in which the SQL statement was run. See monitor element “stmt\_invocation\_id - Statement invocation identifier monitor element” on page 1247 for more details.

**Contained by:** “db2\_activity\_details” on page 158

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **stmt\_source\_id**

This element shows the internal identifier given to the source of the the SQL statement that was run. See monitor element “stmt\_source\_id - Statement source identifier” on page 1253 for more details.

**Contained by:** “db2\_activity\_details” on page 158

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **stmt\_pkgcache\_id**

This element shows the internal package cahce identifier of a dynamic SQL statement. See monitor element “stmt\_pkgcache\_id - Statement package cache identifier monitor element” on page 1251 for more details.

**Contained by:** “db2\_activity\_details” on page 158

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **stmt\_text**

The text of the SQL statement. See monitor element “stmt\_text - SQL statement text monitor element” on page 1255 for more details.

**Contained by:** “db2\_activity\_details” on page 158

### stmt\_first\_use\_time

This element shows the first time the statement entry was processed. For cursor operations, “stmt\_first\_use\_time - Statement first use timestamp monitor element” on page 1246 shows when the cursor was opened. At application coordination nodes, this value reflects the application requests; at non-coordinator nodes, this value reflects when requests were received from the originating node. See monitor element stmt\_first\_use\_time for more details.

**Contained by:** “db2\_activity\_details”

#### Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### stmt\_last\_use\_time

This element shows the last time the statement entry was processed. For cursor operations, “stmt\_last\_use\_time - Statement last use timestamp monitor element” on page 1248 shows the time of the last action on the cursor where that action could be an open, fetch, or close. At application coordination nodes, this value reflects the application requests; at non-coordinator nodes, this value reflects when requests were received from the originating node. See monitor element stmt\_last\_use\_time for more details.

**Contained by:** “db2\_activity\_details”

#### Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### query\_actual\_degree

The actual runtime degree value for the SQL statement while it was being run. See monitor element “query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element” on page 1162 for more details.

**Contained by:** “db2\_activity\_details”

#### Element content:

| Type   | Facet |
|--------|-------|
| xs:int |       |

### db2\_activity\_details

Schema represents the details regarding this activity

**Contained by:** “db2\_activity” on page 148

**Element content:** ( "activity\_id" on page 154, "uow\_id" on page 154, "package\_name" on page 154, "package\_schema" on page 154, "package\_version\_id" on page 154, "consistency\_token" on page 154, "section\_number" on page 154, "reopt" on page 155, "incremental\_bind" on page 155, "effective\_isolation" on page 155, "effective\_query\_degree" on page 155, "stmt\_unicode" on page 155, "stmt\_lock\_timeout" on page 156, "stmt\_type" on page 156, "stmt\_operation" on page 156, "stmt\_query\_id" on page 156, "stmt\_nest\_level" on page 156, "stmt\_invocation\_id" on page 157, "stmt\_source\_id" on page 157, "stmt\_pkgcache\_id" on page 157, "stmt\_text" on page 157, "stmt\_first\_use\_time" on page 158, "stmt\_last\_use\_time" on page 158, "query\_actual\_degree" on page 158, ANY content ( skip ) {zero or more (\*)} )

### db2\_input\_variable

Schema element represents the list of input variables associated with the SQL statement.

**Contained by:** "db2\_activity" on page 148

**Element content:** ( "stmt\_value\_index", "stmt\_value\_isreopt", "stmt\_value\_isnull", "stmt\_value\_type" on page 160, "stmt\_value\_data" on page 160, ANY content ( skip ) {zero or more (\*)} )

### stmt\_value\_index

The element represents the position of the input parameter marker or host variable used in the SQL statement. See monitor element "stmt\_value\_index - Value index" on page 1259 for more details.

**Contained by:** "db2\_input\_variable"

### stmt\_value\_isreopt

The element shows whether the variable was used during statement reoptimization. See monitor element "stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261 for more details.

**Contained by:** "db2\_input\_variable"

### Attributes:

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

### stmt\_value\_isnull

The element shows whether a data value associated with the SQL statement is the NULL value. See monitor element "stmt\_value\_isnull - Value has null value monitor element" on page 1260 for more details.

**Contained by:** "db2\_input\_variable"

### Attributes:

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

### stmt\_value\_type

“stmt\_value\_type - Value type monitor element” on page 1261

**Contained by:** “db2\_input\_variable” on page 159

### stmt\_value\_data

The element contains a string representation of a data value associated with an SQL statement. See monitor element “stmt\_value\_data - Value data” on page 1259 for more details.

**Contained by:** “db2\_input\_variable” on page 159

## Collecting lock event data and generating reports

You can use the lock event monitor to collect lock timeout, lock wait, and deadlock information to help identify and resolve locking problems. After the lock event data has been collected in an unreadable form in an unformatted event table, this task describes how to obtain a readable text report.

### Before you begin

To create the locking event monitor and collect lock event monitor data, you must have DBADM, or SQLADM authority.

### About this task

The lock event monitor collects relevant information that helps with the identification and resolution of locking problems. For example, some of the information the lock event monitor collects for a lock event is as follows:

- The lock that resulted in a lock event
- The applications requesting or holding the lock that resulted in a lock event
- What the applications were doing during the lock event

This task provides instructions for collecting lock event data for a given workload. You might want to collect lock event data under the following conditions:

- You notice that lock wait values are longer than usual when using the MON\_GET\_WORKLOAD table function.
- An application returns a -911 SQL return code with reason code 68 in the administration notification log, stating that “The transaction was rolled back due to a lock timeout.” See also message SQL0911N for further details.
- You notice a deadlock event message in the administration notification log (-911 SQL return code with reason code 2, stating that “The transaction was rolled back due to a deadlock.”). The log message indicates that the lock event occurred between two applications, for example, Application A and B, where A is part of workload FINANCE and B is part of workload PAYROLL. See also message SQL0911N for further details.

Restrictions

To view data values, you need the EXECUTE privilege on the EVMON\_FORMAT\_UE\_\* routines, which the SQLADM and DBADM authorities hold implicitly. You also need SELECT privilege on the unformatted event table, which by default is held by users with the DATAACCESS authority and by the creator of the event monitor and the associated unformatted event table.

## Procedure

To collect detailed information regarding potential future lock events, perform the following steps:

1. Create a lock event monitor called lockevmon by using the CREATE EVENT MONITOR FOR LOCKING statement, as shown in the following example:

```
CREATE EVENT MONITOR lockevmon FOR LOCKING
  WRITE TO UNFORMATTED EVENT TABLE
```

**Note:** The following lists important points to remember when creating an event monitor:

- You can create event monitors ahead of time and not worry about using up disk space since nothing is written until you activate the data collection at the database or workload level
  - In a partitioned database environment, ensure that the event monitors are placed in a partitioned table space across all nodes. Otherwise, lock events will be missed at partitions where the partitioned table space is not present.
  - Ensure that you set up a table space and bufferpool to minimize the interference on high performance work caused by ongoing work during accesses to the tables to obtain data.
2. Activate the lock event monitor called lockevmon by running the following statement:  

```
SET EVENT MONITOR lockevmon STATE 1
```
  3. To enable the lock event data collection at the workload level, issue the ALTER WORKLOAD statement with one of the following COLLECT clauses: COLLECT LOCK TIMEOUT DATA, COLLECT DEADLOCK DATA, or COLLECT LOCK WAIT DATA. Specify the WITH HISTORY option on the COLLECT clause. Setting the database configuration parameter affects the lock event data collection at the database level and all workloads are affected.

### For lock wait events

To collect lock wait data for any lock acquired after 5 seconds for the FINANCE application and to collect lock wait data for any lock acquired after 10 seconds for the PAYROLL application, issue the following statements:

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES
  FOR LOCKS WAITING MORE THAN 5 SECONDS
ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA
  FOR LOCKS WAITING MORE THAN 10 SECONDS WITH HISTORY
```

To set the **mon\_lockwait** database configuration parameter with HIST\_AND\_VALUES input data value for the SAMPLE database, and to set the **mon\_lw\_thresh** database configuration parameter for 10 seconds, issue the following commands:

```
db2 update db cfg for sample using mon_lockwait hist_and_values
db2 update db cfg for sample using mon_lw_thresh 10000000
```

### For lock timeout events

To collect lock timeout data for the FINANCE and PAYROLL applications, issue the following statements:

```
ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA WITH HISTORY
ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

To set the **mon\_locktimeout** database configuration parameter with HIST\_AND\_VALUES input data value for the SAMPLE database, issue the following command:

```
db2 update db cfg for sample using mon_locktimeout hist_and_values
```

#### For deadlock events

To collect data for the FINANCE and PAYROLL applications, issue the following statements:

```
ALTER WORKLOAD finance COLLECT DEADLOCK DATA WITH HISTORY
ALTER WORKLOAD payroll COLLECT DEADLOCK DATA WITH HISTORY
```

To set the **mon\_deadlock** database configuration parameter with HIST\_AND\_VALUES input data value for the SAMPLE database, issue the following command:

```
db2 update db cfg for sample using mon_deadlock hist_and_values
```

4. Rerun the workload in order to receive another lock event notification.
5. Connect to the database.
6. Obtain the locking event report using one of the following approaches:
  - a. Use the XML parser tool, **db2evmonfmt**, to produce a flat-text report based on the event data collected in the unformatted event table and using the default stylesheet, for example:

```
java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
```
  - b. Use the EVMON\_FORMAT\_UE\_TO\_XML table function to obtain an XML document.
  - c. Use the EVMON\_FORMAT\_UE\_TO\_TABLES procedure to output the data into a relational table.
7. Analyze the report to determine the reason for the lock event problem and resolve it.
8. Turn OFF lock data collection for both FINANCE and PAYROLL applications by running the following statements or resetting the database configuration parameters:

#### For lock wait events

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA NONE
```

To reset the **mon\_lockwait** database configuration parameter with the default NONE input data value for the SAMPLE database, and to reset the **mon\_lw\_thresh** database configuration parameter back to its default value of 5 seconds, issue the following command:

```
db2 update db cfg for sample using mon_lockwait none
db2 update db cfg for sample using mon_lw_thresh 5000000
```

#### For lock timeout events

```
ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA NONE
```

To reset the **mon\_locktimeout** database configuration parameter with the default NONE input data value for the SAMPLE database, issue the following command:

```
db2 update db cfg for sample using mon_locktimeout none
```

### For deadlock events

```
ALTER WORKLOAD finance COLLECT DEADLOCK DATA NONE
ALTER WORKLOAD payroll COLLECT DEADLOCK DATA NONE
```

To reset the **mon\_deadlock** database configuration parameter with the default **WITHOUT\_HIST** input data value for the **SAMPLE** database, issue the following command:

```
db2 update db cfg for sample using mon_deadlock without_hist
```

### What to do next

Rerun the application or applications to ensure that the locking problem has been eliminated.

## Unit of work event monitoring

The unit of work event monitor records an event whenever a unit of work is completed, that is, whenever there is a commit or a rollback. This historical information about individual units of work is useful for chargeback purposes (charging by CPU usage) and for monitoring compliance with response-time service-level objectives.

The unit of work event monitor is one way to perform system-perspective monitoring with request metrics. The most closely related alternatives or complements to the unit of work event monitor are the statistics event monitor and the **MON\_GET\_UNIT\_OF\_WORK** and **MON\_GET\_UNIT\_OF\_WORK\_DETAILS** table functions.

You can use the unit of work event monitor to collect a listing of packages used within a unit of work and the nesting level at which it was used. This information helps facilitate stored procedure troubleshooting. Starting in DB2 Version 10.1, you can also generate a listing of the executable IDs and associated statement-level metrics for statements that ran within a unit of work.

To create the unit of work event monitor and collect unit of work event monitor data, you must have **DBADM** or **SQLADM** authority.

### Creating a unit of work event monitor

Starting in DB2 Version 10.1, you have the choice of having the output for a unit of work event monitor output written to an unformatted event (**UE**) table, or a regular table. See “Output options for event monitors” on page 34 for more information on how to choose the most appropriate output format.

Whichever type of table you use, when you create a unit of work event monitor, identify the table space where you plan to store the table or tables containing the output for your event monitor. The recommended practice is to have a table space that is dedicated and configured to store the table. However, when you create an event monitor, you can specify an existing table space. If you do not specify a table space, one is chosen for you.

To create a unit of work event monitor using defaults and best practices, use the **CREATE EVENT MONITOR** statement. The following sample statement uses defaults where possible and specifies that output will be stored in a **UE** table in the **MY\_EVMON\_TABLESPACE** table space:

```
CREATE EVENT MONITOR MY_UOW_EVMON
  FOR UNIT OF WORK
  WRITE TO UNFORMATTED EVENT TABLE (IN MY_EVMON_TABLESPACE)
```

## Configuring data collection

You can specify four different collection levels for unit of work data:

1. None
2. Basic unit of work data
  - a. Information about the packages that ran within the unit of work
  - b. A list of executable IDs for statements that ran within the unit of work.

You can use database configuration parameters to control the collection of unit of work data for all unit of work event monitors that are active in the database. Alternatively, to control the collection of information for specific workloads, service classes, or work actions, you can use the CREATE and ALTER statements for the appropriate workload objects.

To configure data collection at the database level, set the **mon\_uow\_data** database configuration parameter and, optionally, the **mon\_uow\_pkglist** and **mon\_uow\_execlist** database configuration parameters by using the **UPDATE DATABASE CONFIGURATION** command. Possible combinations of values for these parameters are shown in Table 28:

Table 28. Possible values for unit of work event monitor configuration parameters

| Data to collect   | mon_uow_data   | mon_uow_pkglist | mon_uow_execlist |
|---|----------------|-----------------|------------------|
| Collect no unit of work data  | NONE (default) | OFF(default)    | OFF (default)    |
| Collect only basic unit of work data  | BASE           | OFF (default)   | OFF (default)    |
| Collect package list information but not information about executable IDs                       | BASE           | ON              | OFF (default)    |
| Collect information about executable IDs but not a list of packages                             | BASE           | OFF (default)   | ON               |
| Collect basic unit of work data, package list information, and information about executable IDs | BASE           | ON              | ON               |

### Tips:

- If you do not set any of the configuration parameters, no unit of work data is collected unless you enable collection for specific workload objects. You can enable collection for specific workload objects by using either the CREATE or ALTER statement for the appropriate workload object type, for example, the CREATE SERVICE CLASS or ALTER WORKLOAD statement.
- To collect basic unit of work data but no package list or executable ID information, you can set the **mon\_uow\_data** configuration parameter to BASE and omit the **mon\_uow\_pkglist** and **mon\_uow\_execlist** configuration parameters. If you do not explicitly set them, the default value of OFF is used.

- To collect one or both of package list and executable ID information, you must also set the `mon_uow_data` configuration parameter to `BASE`. If you set the `mon_uow_data` configuration parameter to `NONE`, no information is collected, regardless of the settings of the `mon_uow_pkglist` and `mon_uow_execlist` configuration parameters.

To control the collection of data for specific workload objects, use the `COLLECT UNIT OF WORK DATA` clause of the `CREATE` or `ALTER` statement for the specific type of workload object that you are interested in. For example, to collect basic unit of work event data and package list information for the workload `REPORTS`, you might issue a statement such as this:

```
ALTER WORKLOAD REPORTS COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST
```

To collect both package list information and a list of the executable IDs for statements that are run in the unit of work, you might use this statement:

```
ALTER WORKLOAD REPORTS COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST, EXECUTABLE LIST
```

The settings that are shown in Table 28 on page 164 apply to all workloads running in the system unless you override these settings for specific workloads by using the `CREATE WORKLOAD` or `ALTER WORKLOAD` statement. If you want to collect base-level information for all workloads but also want to collect package list information for selected workloads, set the `mon_uow_data` database configuration parameter to `BASE`. Then, use the `CREATE WORKLOAD` or `ALTER WORKLOAD` statement to set the level to `BASE PACKAGE LIST` for the workloads that you are interested in.

By default, applicable table functions and event monitors, including the unit of work event monitor, collect and report request metrics. You can change the default setting as follows:

- By using the `mon_req_metrics` database configuration parameter
- By using the `COLLECT REQUEST METRICS` clause of the `CREATE SERVICE CLASS` or `ALTER SERVICE CLASS` statement for a service superclass.

Changing the default setting affects any table function or event monitor that can report request metrics.

## Accessing event data that is captured by a unit of work event monitor

A unit of work event monitor can write data to a regular table or it can write data in binary format to an unformatted event (UE) table. You can access the data in regular tables by using SQL.

To access data in a UE table, use one of the following table functions:

### **EVMON\_FORMAT\_UE\_TO\_XML**

Extracts data from an unformatted event table into an XML document.

### **EVMON\_FORMAT\_UE\_TO\_TABLES**

Extracts data from an unformatted event table into a set of relational tables.

When you use one of these table functions, you can specify which data to extract by including a `SELECT` statement as one of the parameters to the function. You have full control over selection, ordering, and other aspects provided by the `SELECT` statement.

If you are generating package listing information, you can use the `EVMON_FORMAT_UE_TO_XML` table function to generate a single XML document that contains both the base unit of work event monitor data and the package listing. The `EVMON_FORMAT_UE_TO_TABLES` procedure produces two tables: one for the basic unit of work event monitor information and another for the package listing information. You can join the two tables by using the values in the `MEMBER`, `APPLICATION_ID`, and `UOW_ID` columns.

You can also use the `db2evmonfmt` command to help perform the following tasks:

- Select events of interest based on the following attributes: event ID, event type, time period, application, workload, or service class
- Choose whether to receive the output in the form of a text report or a formatted XML document
- Control the output format by creating your own XSLT style sheets instead of using the ones provided by the `db2evmonfmt` command

For example, the following command provides a unit of work report that selects unit of work events that occurred in the past 24 hours in the database `SAMPLE`. These event records are obtained from the unformatted event table called `SAMPLE_UOW_EVENTS`. The command creates formatted text output by using the `MyUOW.xsl` style sheet.

```
java db2evmonfmt -d SAMPLE -ue SAMPLE_UOW_EVENTS -ftext -ss MyUOW.xsl -hours 24
```

### Data generated by unit of work event monitors

Unit of work event monitors produce data about units of work (transactions) that run in the system. You can choose to have the output from a unit of work event monitor to regular tables, or to an unformatted event (UE) table. If data is written to a UE table, you must perform post-processing on it to view the data.

Regardless of the output format you choose, unit of work event data comes from one of four logical groups:

- `uow`
- `uow_metrics`
- `uow_package_list`
- `uow_exec_list`

If you choose to have the unit of work event data written to regular tables, data from an additional group (`CONTROL`) is used to generate meta-data about the event monitor itself.

**Note:** Unless you specify otherwise, the only monitor elements collected by a unit of work event monitor are request metrics. To enable the generation of basic unit of work event data, or for package or execution list data, you must enable data collection explicitly. See “Enabling event monitor data collection” on page 116 for more information.

### Information written to tables for a unit of work event monitor:

Information written by the unit of work event monitor when the `WRITE TO TABLE` option is specified.

When you choose `WRITE TO TABLE` as the output type for the unit of work event monitor, by default, five tables are produced, each containing monitor elements from one or more logical data groups:

Table 29. Tables produced by UNIT OF WORK write-to-table event monitors. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name                     | Logical data groups reported  |
|--|---|
| UOW_ <i>evmon-name</i>                 | uow   |
| UOW_METRICS_ <i>evmon-name</i>         | uow_metrics   |
| UOW_PACKAGE_LIST_ <i>evmon-name</i>    | uow_package_list  |
| UOW_EXECUTABLE_LIST_ <i>evmon-name</i> | uow_executable_list   |
| CONTROL_ <i>evmon-name</i>             | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

**Note:** Even though all five tables are produced by default, you must still ensure data collection is enabled for the kind of lock information you want to gather, otherwise some of the columns will report null values.

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

### Tables produced

Table 30. Information returned for a locking event monitor: Default table name: UOW\_*evmon-name*

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| APPLICATION_HANDLE | BIGINT       | application_handle - Application handle                         |
| APPLICATION_ID     | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666          |
| APPLICATION_NAME   | VARCHAR(128) | "appl_name - Application name monitor element" on page 670      |
| CLIENT_ACCTNG      | VARCHAR(255) | client_acctng - Client accounting string                        |
| CLIENT_APPLNAME    | VARCHAR(255) | client_applname - Client application name                       |
| CLIENT_HOSTNAME    | VARCHAR(255) | client_hostname - Client hostname                               |
| CLIENT_PID         | BIGINT       | client_pid - Client process ID                                  |
| CLIENT_PLATFORM    | VARCHAR(12)  | client_platform - Client operating platform                     |
| CLIENT_PORT_NUMBER | INTEGER      | client_port_number - Client port number                         |

Table 30. Information returned for a locking event monitor: Default table name: UOW\_evmon-name (continued)

| Column name               | Data type          | Description   |
|---------------------------|--------------------|---|
| CLIENT_PRODUCT_ID         | VARCHAR(128)       | "client_prdid - Client product and version ID monitor element" on page 715          |
| CLIENT_PROTOCOL           | VARCHAR(10)        | client_protocol - Client communication protocol                                     |
| CLIENT_USERID             | VARCHAR(255)       | client_userid - Client user ID  |
| CLIENT_WRKSTNNAME         | VARCHAR(255)       | client_wrkstnname - Client workstation name   |
| COMPLETION_STATUS         | VARCHAR(128)       | completion_status - Completion status   |
| CONNECTION_TIME           | TIMESTAMP          | "connection_start_time - Connection start time monitor element" on page 736         |
| COORD_MEMBER              | SMALLINT           | coord_member - Coordinator member   |
| EVENT_ID                  | INTEGER NOT NULL   | event_id - Event ID monitor element   |
| EVENT_TIMESTAMP           | TIMESTAMP NOT NULL | event_timestamp - Event timestamp monitor element                                   |
| EXECUTABLE_LIST_SIZE      | BIGINT             | "executable_list_size - Size of executable list monitor element" on page 819        |
| EXECUTABLE_LIST_TRUNCATED | CHAR(3)            | "executable_list_truncated - Executable list truncated monitor element" on page 819 |
| GLOBAL_TRANSACTION_ID     | VARCHAR(40)        | "global_transaction_id - Global transaction identifier monitor element" on page 851 |
| INTRA_PARALLEL_STATE      | VARCHAR(128)       | intra_parallel_state - Current state of intrapartition parallelism                  |
| LOCAL_TRANSACTION_ID      | VARCHAR(16)        | "local_transaction_id - Local transaction identifier monitor element" on page 900   |
| MEMBER                    | SMALLINT           | member - Database member  |
| MEMBER_ACTIVATION_TIME    | TIMESTAMP          | "db_conn_time - Database activation timestamp monitor element" on page 772          |
| METRICS                   | BLOB               |   |
| MON_INTERVAL_ID           | VARCHAR(128)       | mon_interval_id - Monitor interval identifier                                       |
| PACKAGE_LIST_EXCEEDED     | CHAR(3)            | package_list_exceeded - Package list exceeded                                       |
| PACKAGE_LIST_SIZE         | INTEGER            | "package_list_size - Size of package list monitor element" on page 1006             |

Table 30. Information returned for a locking event monitor: Default table name: UOW\_evmon-name (continued)

| Column name             | Data type    | Description   |
|-------------------------|--------------|---|
| SERVICE_CLASS_ID        | INTEGER      | service_class_id - Service class ID   |
| SERVICE_SUBCLASS_NAME   | VARCHAR(128) | service_subclass_name - Service subclass name                                   |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name                               |
| SESSION_AUTHID          | VARCHAR(128) | "session_auth_id - Session authorization ID monitor element" on page 1211       |
| START_TIME              | TIMESTAMP    | start_time - Event start time   |
| STOP_TIME               | TIMESTAMP    | stop_time - Event stop time   |
| SYSTEM_AUTHID           | VARCHAR(128) | "system_auth_id - System authorization identifier monitor element" on page 1267 |
| UOW_ID                  | INTEGER      | uow_id - Unit of work ID  |
| UOW_LOG_SPACE_USED      | BIGINT       | uow_log_space_used - Unit of work log space used                                |
| WORKLOAD_ID             | INTEGER      | workload_id - Workload ID   |
| WORKLOAD_NAME           | VARCHAR(128) | workload_name - Workload name   |
| WORKLOAD_OCCURRENCE_ID  | INTEGER      | workload_occurrence_id - Workload occurrence identifier                         |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name

| Column Name                 | Data Type    | Description  |
|-----------------------------|--------------|--|
| PARTITION_KEY               | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016        |
| APPLICATION_ID              | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666                 |
| MEMBER                      | SMALLINT     | member - Database member   |
| UOW_ID                      | INTEGER      | uow_id - Unit of work ID   |
| WLM_QUEUE_TIME_TOTAL        | BIGINT       | wlm_queue_time_total - Workload manager total queue time               |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT       | wlm_queue_assignments_total - Workload manager total queue assignments |
| FCM_TQ_RECV_WAIT_TIME       | BIGINT       | fcm_tq_recv_wait_time - FCM table queue received wait time             |
| FCM_MESSAGE_RECV_WAIT_TIME  | BIGINT       | fcm_message_recv_wait_time - FCM message received wait time            |
| FCM_TQ_SEND_WAIT_TIME       | BIGINT       | fcm_tq_send_wait_time - FCM table queue send wait time                 |
| FCM_MESSAGE_SEND_WAIT_TIME  | BIGINT       | fcm_message_send_wait_time - FCM message send wait time                |
| AGENT_WAIT_TIME             | BIGINT       | agent_wait_time - Agent wait time                                      |
| AGENT_WAITS_TOTAL           | BIGINT       | agent_waits_total - Total agent waits                                  |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                 | Data Type | Description  |
|-----------------------------|-----------|--|
| LOCK_WAIT_TIME              | BIGINT    | lock_wait_time - Time waited on locks                              |
| LOCK_WAITS                  | BIGINT    | lock_waits - Lock waits  |
| DIRECT_READ_TIME            | BIGINT    | direct_read_time - Direct read time                                |
| DIRECT_READ_REQS            | BIGINT    | direct_read_reqs - Direct read requests                            |
| DIRECT_WRITE_TIME           | BIGINT    | direct_write_time - Direct write time                              |
| DIRECT_WRITE_REQS           | BIGINT    | direct_write_reqs - Direct write requests                          |
| LOG_BUFFER_WAIT_TIME        | BIGINT    | log_buffer_wait_time - Log buffer wait time                        |
| NUM_LOG_BUFFER_FULL         | BIGINT    | num_log_buffer_full - Number of full log buffers                   |
| LOG_DISK_WAIT_TIME          | BIGINT    | log_disk_wait_time - Log disk wait time                            |
| LOG_DISK_WAITS_TOTAL        | BIGINT    | log_disk_waits_total - Total log disk waits                        |
| TCPIP_RECV_WAIT_TIME        | BIGINT    | tcPIP_recv_wait_time - TCP/IP received wait time                   |
| TCPIP_RECVS_TOTAL           | BIGINT    | tcPIP_recvs_total - TCP/IP receives total                          |
| CLIENT_IDLE_WAIT_TIME       | BIGINT    | client_idle_wait_time - Client idle wait time                      |
| IPC_RECV_WAIT_TIME          | BIGINT    | ipc_recv_wait_time - Interprocess communication received wait time |
| IPC_RECVS_TOTAL             | BIGINT    | ipc_recvs_total - Interprocess communication receives total        |
| IPC_SEND_WAIT_TIME          | BIGINT    | ipc_send_wait_time - Interprocess communication send wait time     |
| IPC_SENDS_TOTAL             | BIGINT    | ipc_sends_total - Interprocess communication send total            |
| TCPIP_SEND_WAIT_TIME        | BIGINT    | tcPIP_send_wait_time - TCP/IP send wait time                       |
| TCPIP_SENDS_TOTAL           | BIGINT    | tcPIP_sends_total - TCP/IP sends total                             |
| POOL_WRITE_TIME             | BIGINT    | pool_write_time - Total buffer pool physical write time            |
| POOL_READ_TIME              | BIGINT    | pool_read_time - Total buffer pool physical read time              |
| AUDIT_FILE_WRITE_WAIT_TIME  | BIGINT    | audit_file_write_wait_time - Audit file write wait time            |
| AUDIT_FILE_WRITES_TOTAL     | BIGINT    | audit_file_writes_total - Total audit files written                |
| AUDIT_SUBSYSTEM_WAIT_TIME   | BIGINT    | audit_subsystem_wait_time - Audit subsystem wait time              |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT    | audit_subsystem_waits_total - Total audit subsystem waits          |
| DIAGLOG_WRITE_WAIT_TIME     | BIGINT    | diaglog_write_wait_time - Diagnostic log file write wait time      |
| DIAGLOG_WRITES_TOTAL        | BIGINT    | diaglog_writes_total - Total diagnostic log file writes            |
| FCM_SEND_WAIT_TIME          | BIGINT    | fcm_send_wait_time - FCM send wait time                            |
| FCM_RECV_WAIT_TIME          | BIGINT    | fcm_recv_wait_time - FCM received wait time                        |
| TOTAL_WAIT_TIME             | BIGINT    | total_wait_time - Total wait time                                  |
| RQSTS_COMPLETED_TOTAL       | BIGINT    | rqsts_completed_total - Total requests completed                   |
| TOTAL_RQST_TIME             | BIGINT    | total_rqst_time - Total request time                               |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                  | Data Type | Description   |
|------------------------------|-----------|---|
| APP_RQSTS_COMPLETED_TOTAL    | BIGINT    | app_rqsts_completed_total - Total application requests completed      |
| TOTAL_APP_RQST_TIME          | BIGINT    | total_app_rqst_time - Total application request time                  |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT    | total_section_sort_proc_time - Total section sort processing time     |
| TOTAL_SECTION_SORTS          | BIGINT    | total_section_sorts - Total section sorts                             |
| TOTAL_SECTION_SORT_TIME      | BIGINT    | total_section_sort_time - Total section sort time                     |
| ROWS_READ                    | BIGINT    | rows_read - Rows read   |
| ROWS_MODIFIED                | BIGINT    | rows_modified - Rows modified   |
| POOL_DATA_L_READS            | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                    |
| POOL_INDEX_L_READS           | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_TEMP_DATA_L_READS       | BIGINT    | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_INDEX_L_READS      | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_XDA_L_READS             | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_TEMP_XDA_L_READS        | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| TOTAL_CPU_TIME               | BIGINT    | total_cpu_time - Total CPU time                                       |
| ACT_COMPLETED_TOTAL          | BIGINT    | act_completed_total - Total completed activities                      |
| POOL_DATA_P_READS            | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_TEMP_DATA_P_READS       | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_XDA_P_READS             | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_TEMP_XDA_P_READS        | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS           | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_TEMP_INDEX_P_READS      | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_DATA_WRITES             | BIGINT    | pool_data_writes - Buffer pool data writes                            |
| POOL_XDA_WRITES              | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| POOL_INDEX_WRITES            | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| DIRECT_READS                 | BIGINT    | direct_reads - Direct reads from database                             |
| DIRECT_WRITES                | BIGINT    | direct_writes - Direct writes to database                             |
| ROWS_RETURNED                | BIGINT    | rows_returned - Rows returned   |
| DEADLOCKS                    | BIGINT    | deadlocks - Deadlocks detected  |
| LOCK_TIMEOUTS                | BIGINT    | lock_timeouts - Number of lock timeouts                               |
| LOCK_ESCALS                  | BIGINT    | lock_escals - Number of lock escalations                              |
| FCM_SENDS_TOTAL              | BIGINT    | fcm_sends_total - FCM sends total                                     |
| FCM_RECVS_TOTAL              | BIGINT    | fcm_recvs_total - FCM receives total                                  |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                 | Data Type | Description   |
|-----------------------------|-----------|---|
| FCM_SEND_VOLUME             | BIGINT    | fcm_send_volume - FCM send volume                                   |
| FCM_RECV_VOLUME             | BIGINT    | fcm_recv_volume - FCM received volume                               |
| FCM_MESSAGE_SENDS_TOTAL     | BIGINT    | fcm_message_sends_total - Total FCM message sends                   |
| FCM_MESSAGE_RECVS_TOTAL     | BIGINT    | fcm_message_recvs_total - Total FCM message receives                |
| FCM_MESSAGE_SEND_VOLUME     | BIGINT    | fcm_message_send_volume - FCM message send volume                   |
| FCM_MESSAGE_RECV_VOLUME     | BIGINT    | fcm_message_recv_volume - FCM message received volume               |
| FCM_TQ_SENDS_TOTAL          | BIGINT    | fcm_tq_sends_total - FCM table queue send total                     |
| FCM_TQ_RECVS_TOTAL          | BIGINT    | fcm_tq_recvs_total - FCM table queue receives total                 |
| FCM_TQ_SEND_VOLUME          | BIGINT    | fcm_tq_send_volume - FCM table queue send volume                    |
| FCM_TQ_RECV_VOLUME          | BIGINT    | fcm_tq_recv_volume - FCM table queue received volume                |
| TQ_TOT_SEND_SPILLS          | BIGINT    | tq_tot_send_spills - Total number of table queue buffers overflowed |
| TCPIP_SEND_VOLUME           | BIGINT    | tcPIP_send_volume - TCP/IP send volume                              |
| TCPIP_RECV_VOLUME           | BIGINT    | tcPIP_recv_volume - TCP/IP received volume                          |
| IPC_SEND_VOLUME             | BIGINT    | ipc_send_volume - Interprocess communication send volume            |
| IPC_RECV_VOLUME             | BIGINT    | ipc_recv_volume - Interprocess communication received volume        |
| POST_THRESHOLD_SORTS        | BIGINT    | post_threshold_sorts - Post threshold sorts                         |
| POST_SHRTHRESHOLD_SORTS     | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts               |
| SORT_OVERFLOWS              | BIGINT    | sort_overflows - Sort overflows                                     |
| AUDIT_EVENTS_TOTAL          | BIGINT    | audit_events_total - Total audit events                             |
| ACT_REJECTED_TOTAL          | BIGINT    | act_rejected_total - Total rejected activities                      |
| ACT_ABORTED_TOTAL           | BIGINT    | act_aborted_total - Total aborted activities                        |
| TOTAL_SORTS                 | BIGINT    | total_sorts - Total sorts   |
| TOTAL_ROUTINE_TIME          | BIGINT    | total_routine_time - Total routine time                             |
| TOTAL_COMPILE_PROC_TIME     | BIGINT    | total_compile_proc_time - Total compile processing time             |
| TOTAL_COMPILATIONS          | BIGINT    | total_compilations - Total compilations                             |
| TOTAL_COMPILE_TIME          | BIGINT    | total_compile_time - Total compile time                             |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT    | total_implicit_compilations - Total implicit complications          |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT    | total_implicit_compile_time - Total implicit compile time           |
| TOTAL_RUNSTATS_PROC_TIME    | BIGINT    | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS              | BIGINT    | total_runstats - Total runtime statistics                           |
| TOTAL_RUNSTATS_TIME         | BIGINT    | total_runstats_time - Total runtime statistics time                 |
| TOTAL_REORG_PROC_TIME       | BIGINT    | total_reorg_proc_time - Total reorganization processing time        |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                    | Data Type | Description   |
|--------------------------------|-----------|---|
| TOTAL_REORGS                   | BIGINT    | total_reorgs - Total reorganizations                                |
| TOTAL_REORG_TIME               | BIGINT    | total_reorg_time - Total reorganization time                        |
| TOTAL_LOAD_PROC_TIME           | BIGINT    | total_load_proc_time - Total load processing time                   |
| TOTAL_LOADS                    | BIGINT    | total_loads - Total loads   |
| TOTAL_LOAD_TIME                | BIGINT    | total_load_time - Total load time                                   |
| TOTAL_SECTION_PROC_TIME        | BIGINT    | total_section_proc_time - Total section processing time             |
| TOTAL_APP_SECTION_EXECUTIONS   | BIGINT    | total_app_section_executions - Total application section executions |
| TOTAL_SECTION_TIME             | BIGINT    | total_section_time - Total section time                             |
| TOTAL_COMMIT_PROC_TIME         | BIGINT    | total_commit_proc_time - Total commits processing time              |
| TOTAL_APP_COMMITS              | BIGINT    | total_app_commits - Total application commits                       |
| TOTAL_COMMIT_TIME              | BIGINT    | total_commit_time - Total commit time                               |
| TOTAL_ROLLBACK_PROC_TIME       | BIGINT    | total_rollback_proc_time - Total rollback processing time           |
| TOTAL_APP_ROLLBACKS            | BIGINT    | total_app_rollbacks - Total application rollbacks                   |
| TOTAL_ROLLBACK_TIME            | BIGINT    | total_rollback_time - Total rollback time                           |
| TOTAL_ROUTINE_USER_CODE_TIME   | BIGINT    | total_routine_user_code_time - Total routine user code time         |
| THRESH_VIOLATIONS              | BIGINT    | thresh_violations - Number of threshold violations                  |
| NUM_LW_THRESH_EXCEEDED         | BIGINT    | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded    |
| TOTAL_ROUTINE_INVOCATIONS      | BIGINT    | total_routine_invocations - Total routine invocations               |
| INT_COMMITS                    | BIGINT    | int_commits - Internal commits                                      |
| INT_ROLLBACKS                  | BIGINT    | int_rollbacks - Internal rollbacks                                  |
| CAT_CACHE_INSERTS              | BIGINT    | cat_cache_inserts - Catalog cache inserts                           |
| CAT_CACHE_LOOKUPS              | BIGINT    | cat_cache_lookups - Catalog cache lookups                           |
| PKG_CACHE_INSERTS              | BIGINT    | pkg_cache_inserts - Package cache inserts                           |
| PKG_CACHE_LOOKUPS              | BIGINT    | pkg_cache_lookups - Package cache lookups                           |
| ACT_RQSTS_TOTAL                | BIGINT    | act_rqsts_total - Total activity requests                           |
| TOTAL_ACT_WAIT_TIME            | BIGINT    | total_act_wait_time - Total activity wait time                      |
| TOTAL_ACT_TIME                 | BIGINT    | total_act_time - Total activity time                                |
| LOCK_WAIT_TIME_GLOBAL          | BIGINT    | lock_wait_time_global - Lock wait time global                       |
| LOCK_WAITS_GLOBAL              | BIGINT    | lock_waits_global - Lock waits global                               |
| RECLAIM_WAIT_TIME              | BIGINT    | reclaim_wait_time - Reclaim wait time                               |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT    | spacemappage_reclaim_wait_time - Space map page reclaim wait time   |
| LOCK_TIMEOUTS_GLOBAL           | BIGINT    | lock_timeouts_global - Lock timeouts global                         |
| LOCK_ESCALS_MAXLOCKS           | BIGINT    | lock_escals_maxlocks - Number of maxlocks lock escalations          |
| LOCK_ESCALS_LOCKLIST           | BIGINT    | lock_escals_locklist - Number of locklist lock escalations          |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                    | Data Type | Description  |
|--------------------------------|-----------|--|
| LOCK_ESCALS_GLOBAL             | BIGINT    | lock_escals_global - Number of global lock escalations                     |
| CF_WAIT_TIME                   | BIGINT    | cf_wait_time - cluster caching facility wait time                          |
| CF_WAITS                       | BIGINT    | cf_waits - Number of cluster caching facility DB2 pureScale server waits   |
| POOL_DATA_GBP_L_READS          | BIGINT    | pool_data_gbp_l_reads - Group buffer pool data logical reads               |
| POOL_DATA_GBP_P_READS          | BIGINT    | pool_data_gbp_p_reads - Group buffer pool data physical reads              |
| POOL_DATA_LBP_PAGES_FOUND      | BIGINT    | pool_data_lbp_pages_found - Local buffer pool found data pages             |
| POOL_DATA_GBP_INVALID_PAGES    | BIGINT    | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages         |
| POOL_INDEX_GBP_L_READS         | BIGINT    | pool_index_gbp_l_reads - Group buffer pool index logical reads             |
| POOL_INDEX_GBP_P_READS         | BIGINT    | pool_index_gbp_p_reads - Group buffer pool index physical reads            |
| POOL_INDEX_LBP_PAGES_FOUND     | BIGINT    | pool_index_lbp_pages_found - Local buffer pool index pages found           |
| POOL_INDEX_GBP_INVALID_PAGES   | BIGINT    | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages       |
| POOL_XDA_GBP_L_READS           | BIGINT    | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests    |
| POOL_XDA_GBP_P_READS           | BIGINT    | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests   |
| POOL_XDA_LBP_PAGES_FOUND       | BIGINT    | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found          |
| POOL_XDA_GBP_INVALID_PAGES     | BIGINT    | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages      |
| EVMON_WAIT_TIME                | BIGINT    | evmon_wait_time - Event monitor wait time                                  |
| EVMON_WAITS_TOTAL              | BIGINT    | evmon_waits_total - Event monitor total waits                              |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT    | total_extended_latch_wait_time - Total extended latch wait time            |
| TOTAL_EXTENDED_LATCH_WAITS     | BIGINT    | total_extended_latch_waits - Total extended latch waits                    |
| TOTAL_STATS_FABRICATIONS       | BIGINT    | total_stats_fabrications - Total statistics fabrications                   |
| TOTAL_STATS_FABRICATION_TIME   | BIGINT    | total_stats_fabrication_time - Total statistics fabrication time           |
| TOTAL_SYNC_RUNSTATS_PROC_TIME  | BIGINT    | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| TOTAL_SYNC_RUNSTATS            | BIGINT    | total_sync_runstats - Total synchronous RUNSTATS activities                |
| TOTAL_SYNC_RUNSTATS_TIME       | BIGINT    | total_sync_runstats_time - Total synchronous RUNSTATS time                 |
| TOTAL_DISP_RUN_QUEUE_TIME      | BIGINT    | total_disp_run_queue_time - Total dispatcher run queue time                |
| POOL_QUEUED_ASYNC_DATA_REQS    | BIGINT    | pool_queued_async_data_reqs - Data prefetch requests                       |
| POOL_QUEUED_ASYNC_INDEX_REQS   | BIGINT    | pool_queued_async_index_reqs - Index prefetch requests                     |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                             | Data Type | Description   |
|---|-----------|---|
| POOL_QUEUED_ASYNC_XDA_REQS              | BIGINT    | pool_queued_async_xda_reqs - XDA prefetch requests  |
| POOL_QUEUED_ASYNC_OTHER_REQS            | BIGINT    | pool_queued_async_other_reqs - Non-prefetch requests  |
| POOL_QUEUED_ASYNC_DATA_PAGES            | BIGINT    | pool_queued_async_data_pages - Data pages prefetch requests   |
| POOL_QUEUED_ASYNC_INDEX_PAGES           | BIGINT    | pool_queued_async_index_pages - Index pages prefetch requests   |
| POOL_QUEUED_ASYNC_XDA_PAGES             | BIGINT    | pool_queued_async_xda_pages - XDA pages prefetch requests   |
| POOL_FAILED_ASYNC_DATA_REQS             | BIGINT    | pool_failed_async_data_reqs - Failed data prefetch requests   |
| POOL_FAILED_ASYNC_INDEX_REQS            | BIGINT    | pool_failed_async_index_reqs - Failed index prefetch requests   |
| POOL_FAILED_ASYNC_XDA_REQS              | BIGINT    | pool_failed_async_xda_reqs - Failed XDA prefetch requests   |
| POOL_FAILED_ASYNC_OTHER_REQS            | BIGINT    | pool_failed_async_other_reqs - Failed non-prefetch requests   |
| APP_ACT_COMPLETED_TOTAL                 | BIGINT    | app_act_completed_total - Total successful external coordinator activities  |
| APP_ACT_ABORTED_TOTAL                   | BIGINT    | app_act_aborted_total - Total failed external coordinator activities  |
| APP_ACT_REJECTED_TOTAL                  | BIGINT    | app_act_rejected_total - Total rejected external coordinator activities   |
| TOTAL_PEDS                              | BIGINT    | total_peds - Total partial early distincts  |
| DISABLED_PEDS                           | BIGINT    | “disabled_peds - Disabled partial early distincts monitor element” on page 802  |
| POST_THRESHOLD_PEDS                     | BIGINT    | post_threshold_peds - Partial early distincts threshold   |
| TOTAL_PEAS                              | BIGINT    | total_peas - Total partial early aggregations   |
| POST_THRESHOLD_PEAS                     | BIGINT    | post_threshold_peas - Partial early aggregation threshold   |
| TQ_SORT_HEAP_REQUESTS                   | BIGINT    | tq_sort_heap_requests - Table queue sort heap requests  |
| TQ_SORT_HEAP_REJECTIONS                 | BIGINT    | tq_sort_heap_rejections - Table queue sort heap rejections  |
| TOTAL_CONNECT_REQUESTS                  | BIGINT    | total_connect_requests - Connection or switch user requests   |
| TOTAL_CONNECT_REQUEST_TIME              | BIGINT    | total_connect_request_time - Total connection or switch user request time   |
| TOTAL_CONNECT_AUTHENTICATIONS           | BIGINT    | total_connect_authentications - Connections or switch user authentications performed  |
| PREFETCH_WAIT_TIME                      | BIGINT    | prefetch_wait_time - Time waited for prefetch   |
| PREFETCH_WAITS                          | BIGINT    | prefetch_waits - Prefetcher wait count  |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP  | BIGINT    | “pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element” on page 1041   |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | “pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element” on page 1073 |

Table 31. Information returned for a unit of work event monitor: Table name: UOW\_METRICS\_evmon-name (continued)

| Column Name                           | Data Type | Description   |
|---------------------------------------|-----------|---|
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | "pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element" on page 1130 |

Table 32. Information returned for a locking event monitor: Default table name: UOW\_PACKAGE\_LIST\_evmon-name

| Column name          | Data type    | Description   |
|----------------------|--------------|---|
| PARTITION_KEY        | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| APPLICATION_ID       | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666          |
| INVOCATION_ID        | INTEGER      | invocation_id - Invocation ID                                   |
| MEMBER               | SMALLINT     | member - Database member  |
| NESTING_LEVEL        | INTEGER      | nesting_level - Nesting level                                   |
| PACKAGE_ELAPSED_TIME | BIGINT       | package_elapsed_time - Package elapsed time                     |
| PACKAGE_ID           | BIGINT       | package_id - Package identifier                                 |
| ROUTINE_ID           | INTEGER      | routine_id - Routine ID   |
| UOW_ID               | INTEGER      | uow_id - Unit of work ID  |

Table 33. Information returned for a locking event monitor: Default table name: UOW\_EXECUTABLE\_LIST\_evmon-name

| Column name             | Data type    | Description   |
|-------------------------|--------------|---|
| PARTITION_KEY           | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| APPLICATION_ID          | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666          |
| EXECUTABLE_ID           | VARCHAR(32)  | executable_id - Executable ID                                   |
| LOCK_WAIT_TIME          | BIGINT       | lock_wait_time - Time waited on locks                           |
| LOCK_WAITS              | BIGINT       | lock_waits - Lock waits   |
| MEMBER                  | SMALLINT     | member - Database member  |
| NUM_EXECUTIONS          | BIGINT       | num_executions - Statement executions                           |
| POST_SHRTHRESHOLD_SORTS | BIGINT       | post_shrthreshold_sorts - Post shared threshold sorts           |
| POST_THRESHOLD_SORTS    | BIGINT       | post_threshold_sorts - Post threshold sorts                     |
| ROWS_READ               | BIGINT       | rows_read - Rows read   |
| SORT_OVERFLOWS          | BIGINT       | sort_overflows - Sort overflows                                 |

Table 33. Information returned for a locking event monitor: Default table name: UOW\_EXECUTABLE\_LIST\_evmon-name (continued)

| Column name         | Data type | Description                                    |
|---------------------|-----------|--|
| TOTAL_ACT_TIME      | BIGINT    | total_act_time - Total activity time           |
| TOTAL_ACT_WAIT_TIME | BIGINT    | total_act_wait_time - Total activity wait time |
| TOTAL_CPU_TIME      | BIGINT    | total_cpu_time - Total CPU time                |
| TOTAL_SORTS         | BIGINT    | total_sorts - Total sorts                      |
| UOW_ID              | INTEGER   | uow_id - Unit of work ID                       |

Table 34. Information returned for a unit of work event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

**Information written to relational tables by EVMON\_FORMAT\_UE\_TO\_TABLES for a unit of work event monitor:**

Information written for a unit of work event monitor from the EVMON\_FORMAT\_UE\_TO\_TABLES table function. This is also documented in the sql1lib/misc/DB2EvmonUOW.xsd file.

Table 35. Information returned for a unit of work event monitor: Table name: UOW\_EVENT

| Column Name       | Data Type             | Description                                       |
|-------------------|-----------------------|---|
| EVENT_ID          | BIGINT NOT NULL       | event_id - Event ID monitor element               |
| TYPE              | VARCHAR(128) NOT NULL |   |
| EVENT_TIMESTAMP   | TIMESTAMP NOT NULL    | event_timestamp - Event timestamp monitor element |
| MEMBER            | SMALLINT              | member - Database member                          |
| COORD_MEMBER      | SMALLINT              | coord_member - Coordinator member                 |
| COMPLETION_STATUS | VARCHAR(128)          | completion_status - Completion status             |
| START_TIME        | TIMESTAMP             | start_time - Event start time                     |
| STOP_TIME         | TIMESTAMP             | stop_time - Event stop time                       |

Table 35. Information returned for a unit of work event monitor: Table name: UOW\_EVENT (continued)

| Column Name             | Data Type    | Description   |
|-------------------------|--------------|---|
| WORKLOAD_NAME           | VARCHAR(128) | workload_name - Workload name   |
| WORKLOAD_ID             | INTEGER      | workload_id - Workload ID   |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name                               |
| SERVICE_SUBCLASS_NAME   | VARCHAR(128) | service_subclass_name - Service subclass name                                   |
| SERVICE_CLASS_ID        | INTEGER      | service_class_id - Service class ID   |
| UOW_ID                  | INTEGER      | uow_id - Unit of work ID  |
| WORKLOAD_OCCURRENCE_ID  | INTEGER      | workload_occurrence_id - Workload occurrence identifier                         |
| CONNECTION_TIME         | TIMESTAMP    | "connection_start_time - Connection start time monitor element" on page 736     |
| MEMBER_ACTIVATION_TIME  | TIMESTAMP    | "db_conn_time - Database activation timestamp monitor element" on page 772      |
| APPLICATION_ID          | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666                          |
| APPLICATION_HANDLE      | BIGINT       | application_handle - Application handle   |
| APPLICATION_NAME        | VARCHAR(128) | "appl_name - Application name monitor element" on page 670                      |
| SYSTEM_AUTHID           | VARCHAR(128) | "system_auth_id - System authorization identifier monitor element" on page 1267 |
| SESSION_AUTHID          | VARCHAR(128) | "session_auth_id - Session authorization ID monitor element" on page 1211       |
| CLIENT_PLATFORM         | VARCHAR(12)  | client_platform - Client operating platform                                     |
| CLIENT_PID              | BIGINT       | client_pid - Client process ID  |
| CLIENT_PRODUCT_ID       | VARCHAR(128) | "client_prdid - Client product and version ID monitor element" on page 715      |
| CLIENT_PROTOCOL         | VARCHAR(10)  | client_protocol - Client communication protocol                                 |
| CLIENT_HOSTNAME         | VARCHAR(255) | client_hostname - Client hostname   |
| CLIENT_PORT_NUMBER      | INTEGER      | client_port_number - Client port number   |
| CLIENT_WRKSTNNAME       | VARCHAR(255) | client_wrkstnname - Client workstation name                                     |
| CLIENT_ACCTNG           | VARCHAR(255) | client_acctng - Client accounting string  |
| CLIENT_USERID           | VARCHAR(255) | client_userid - Client user ID  |
| CLIENT_APPLNAME         | VARCHAR(255) | client_applname - Client application name                                       |

Table 35. Information returned for a unit of work event monitor: Table name: UOW\_EVENT (continued)

| Column Name               | Data Type   | Description   |
|---------------------------|-------------|---|
| LOCAL_TRANSACTION_ID      | VARCHAR(16) | "local_transaction_id - Local transaction identifier monitor element" on page 900   |
| GLOBAL_TRANSACTION_ID     | VARCHAR(40) | "global_transaction_id - Global transaction identifier monitor element" on page 851   |
| UOW_LOG_SPACE_USED        | BIGINT      | uow_log_space_used - Unit of work log space used  |
| PACKAGE_LIST_SIZE         | INTEGER     | "package_list_size - Size of package list monitor element" on page 1006   |
| PACKAGE_LIST_EXCEEDED     | CHAR(3)     | package_list_exceeded - Package list exceeded   |
| EXECUTABLE_LIST_SIZE      | BIGINT      | "executable_list_size - Size of executable list monitor element" on page 819  |
| EXECUTABLE_LIST_TRUNCATED | CHAR(3)     | "executable_list_truncated - Executable list truncated monitor element" on page 819   |
| METRICS                   | BLOB(1M)    | XML document containing metrics-related monitor elements. The metrics in this document are the same as those described in the UOW_METRICS table that appears later in this topic. See "Interfaces that return monitor data in XML documents" on page 13 for more information. |
| INTRA_PARALLEL_STATE      | VARCHAR(3)  | intra_parallel_state - Current state of intrapartition parallelism  |
| MON_INTERVAL_ID           | BIGINT      | mon_interval_id - Monitor interval identifier   |

Table 36. Information returned for a unit of work event monitor: Table name: UOW\_PACKAGE\_LIST

| Column Name          | Data Type    | Description  |
|----------------------|--------------|--|
| MEMBER               | SMALLINT     | member - Database member                               |
| UOW_ID               | INTEGER      | uow_id - Unit of work ID                               |
| APPLICATION_ID       | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666 |
| PACKAGE_ID           | BIGINT       | package_id - Package identifier                        |
| NESTING_LEVEL        | INTEGER      | nesting_level - Nesting level                          |
| ROUTINE_ID           | BIGINT       | routine_id - Routine ID                                |
| INVOCATION_ID        | INTEGER      | invocation_id - Invocation ID                          |
| PACKAGE_ELAPSED_TIME | BIGINT       | package_elapsed_time - Package elapsed time            |

Table 37. Information returned for a unit of work event monitor: Table name: UOW\_EXECUTABLE\_LIST

| Column Name             | Data Type                | Description  |
|-------------------------|--------------------------|--|
| MEMBER                  | SMALLINT                 | member - Database member                               |
| UOW_ID                  | INTEGER                  | uow_id - Unit of work ID                               |
| APPLICATION_ID          | VARCHAR(128)             | "appl_id - Application ID monitor element" on page 666 |
| EXECUTABLE_ID           | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID                          |
| NUM_EXECUTIONS          | BIGINT                   | num_executions - Statement executions                  |
| ROWS_READ               | BIGINT                   | rows_read - Rows read                                  |
| TOTAL_CPU_TIME          | BIGINT                   | total_cpu_time - Total CPU time                        |
| TOTAL_ACT_TIME          | BIGINT                   | total_act_time - Total activity time                   |
| TOTAL_ACT_WAIT_TIME     | BIGINT                   | total_act_wait_time - Total activity wait time         |
| LOCK_WAIT_TIME          | BIGINT                   | lock_wait_time - Time waited on locks                  |
| LOCK_WAITS              | BIGINT                   | lock_waits - Lock waits                                |
| TOTAL_SORTS             | BIGINT                   | total_sorts - Total sorts                              |
| POST_THRESHOLD_SORTS    | BIGINT                   | post_threshold_sorts - Post threshold sorts            |
| POST_SHRTHRESHOLD_SORTS | BIGINT                   | post_shrthreshold_sorts - Post shared threshold sorts  |
| SORT_OVERFLOWS          | BIGINT                   | sort_overflows - Sort overflows                        |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table

| Column Name             | Data Type    | Description   |
|-------------------------|--------------|---|
| MEMBER                  | SMALLINT     | member - Database member  |
| UOW_ID                  | INTEGER      | uow_id - Unit of work ID  |
| APPLICATION_ID          | VARCHAR(128) | "appl_id - Application ID monitor element" on page 666              |
| ACT_ABORTED_TOTAL       | BIGINT       | act_aborted_total - Total aborted activities                        |
| ACT_COMPLETED_TOTAL     | BIGINT       | act_completed_total - Total completed activities                    |
| ACT_REJECTED_TOTAL      | BIGINT       | act_rejected_total - Total rejected activities                      |
| AGENT_WAIT_TIME         | BIGINT       | agent_wait_time - Agent wait time                                   |
| AGENT_WAITS_TOTAL       | BIGINT       | agent_waits_total - Total agent waits                               |
| POOL_DATA_L_READS       | BIGINT       | pool_data_l_reads - Buffer pool data logical reads                  |
| POOL_INDEX_L_READS      | BIGINT       | pool_index_l_reads - Buffer pool index logical reads                |
| POOL_TEMP_DATA_L_READS  | BIGINT       | pool_temp_data_l_reads - Buffer pool temporary data logical reads   |
| POOL_TEMP_INDEX_L_READS | BIGINT       | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name             | Data Type | Description   |
|-------------------------|-----------|---|
| POOL_TEMP_XDA_L_READS   | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| POOL_XDA_L_READS        | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_DATA_P_READS       | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_INDEX_P_READS      | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_TEMP_DATA_P_READS  | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_TEMP_INDEX_P_READS | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_TEMP_XDA_P_READS   | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_XDA_P_READS        | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_DATA_WRITES        | BIGINT    | pool_data_writes - Buffer pool data writes                            |
| POOL_INDEX_WRITES       | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| POOL_XDA_WRITES         | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| POOL_READ_TIME          | BIGINT    | pool_read_time - Total buffer pool physical read time                 |
| POOL_WRITE_TIME         | BIGINT    | pool_write_time - Total buffer pool physical write time               |
| CLIENT_IDLE_WAIT_TIME   | BIGINT    | client_idle_wait_time - Client idle wait time                         |
| DEADLOCKS               | BIGINT    | deadlocks - Deadlocks detected  |
| DIRECT_READS            | BIGINT    | direct_reads - Direct reads from database                             |
| DIRECT_READ_TIME        | BIGINT    | direct_read_time - Direct read time                                   |
| DIRECT_WRITES           | BIGINT    | direct_writes - Direct writes to database                             |
| DIRECT_WRITE_TIME       | BIGINT    | direct_write_time - Direct write time                                 |
| DIRECT_READ_REQS        | BIGINT    | direct_read_reqs - Direct read requests                               |
| DIRECT_WRITE_REQS       | BIGINT    | direct_write_reqs - Direct write requests                             |
| FCM_RECV_VOLUME         | BIGINT    | fcm_recv_volume - FCM received volume                                 |
| FCM_RECVS_TOTAL         | BIGINT    | fcm_recvs_total - FCM receives total                                  |
| FCM_SEND_VOLUME         | BIGINT    | fcm_send_volume - FCM send volume                                     |
| FCM_SENDS_TOTAL         | BIGINT    | fcm_sends_total - FCM sends total                                     |
| FCM_RECV_WAIT_TIME      | BIGINT    | fcm_recv_wait_time - FCM received wait time                           |
| FCM_SEND_WAIT_TIME      | BIGINT    | fcm_send_wait_time - FCM send wait time                               |
| IPC_RECV_VOLUME         | BIGINT    | ipc_recv_volume - Interprocess communication received volume          |
| IPC_RECV_WAIT_TIME      | BIGINT    | ipc_recv_wait_time - Interprocess communication received wait time    |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                 | Data Type | Description  |
|-----------------------------|-----------|--|
| IPC_RECVS_TOTAL             | BIGINT    | ipc_recvs_total - Interprocess communication receives total            |
| IPC_SEND_VOLUME             | BIGINT    | ipc_send_volume - Interprocess communication send volume               |
| IPC_SEND_WAIT_TIME          | BIGINT    | ipc_send_wait_time - Interprocess communication send wait time         |
| IPC_SENDS_TOTAL             | BIGINT    | ipc_sends_total - Interprocess communication send total                |
| LOCK_ESCALS                 | BIGINT    | lock_escals - Number of lock escalations                               |
| LOCK_TIMEOUTS               | BIGINT    | lock_timeouts - Number of lock timeouts                                |
| LOCK_WAIT_TIME              | BIGINT    | lock_wait_time - Time waited on locks                                  |
| LOCK_WAITS                  | BIGINT    | lock_waits - Lock waits  |
| LOG_BUFFER_WAIT_TIME        | BIGINT    | log_buffer_wait_time - Log buffer wait time                            |
| NUM_LOG_BUFFER_FULL         | BIGINT    | num_log_buffer_full - Number of full log buffers                       |
| LOG_DISK_WAIT_TIME          | BIGINT    | log_disk_wait_time - Log disk wait time                                |
| LOG_DISK_WAITS_TOTAL        | BIGINT    | log_disk_waits_total - Total log disk waits                            |
| RQSTS_COMPLETED_TOTAL       | BIGINT    | rqsts_completed_total - Total requests completed                       |
| ROWS_MODIFIED               | BIGINT    | rows_modified - Rows modified  |
| ROWS_READ                   | BIGINT    | rows_read - Rows read  |
| ROWS_RETURNED               | BIGINT    | rows_returned - Rows returned  |
| TCPIP_RECV_VOLUME           | BIGINT    | tcpip_recv_volume - TCP/IP received volume                             |
| TCPIP_SEND_VOLUME           | BIGINT    | tcpip_send_volume - TCP/IP send volume                                 |
| TCPIP_RECV_WAIT_TIME        | BIGINT    | tcpip_recv_wait_time - TCP/IP received wait time                       |
| TCPIP_RECVS_TOTAL           | BIGINT    | tcpip_recvs_total - TCP/IP receives total                              |
| TCPIP_SEND_WAIT_TIME        | BIGINT    | tcpip_send_wait_time - TCP/IP send wait time                           |
| TCPIP_SENDS_TOTAL           | BIGINT    | tcpip_sends_total - TCP/IP sends total                                 |
| TOTAL_APP_RQST_TIME         | BIGINT    | total_app_rqst_time - Total application request time                   |
| TOTAL_RQST_TIME             | BIGINT    | total_rqst_time - Total request time                                   |
| WLM_QUEUE_TIME_TOTAL        | BIGINT    | wlm_queue_time_total - Workload manager total queue time               |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT    | wlm_queue_assignments_total - Workload manager total queue assignments |
| TOTAL_CPU_TIME              | BIGINT    | total_cpu_time - Total CPU time  |
| TOTAL_WAIT_TIME             | BIGINT    | total_wait_time - Total wait time                                      |
| APP_RQSTS_COMPLETED_TOTAL   | BIGINT    | app_rqsts_completed_total - Total application requests completed       |
| TOTAL_SECTION_SORT_TIME     | BIGINT    | total_section_sort_time - Total section sort time                      |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                      | Data Type | Description   |
|----------------------------------|-----------|---|
| TOTAL_SECTION_SORT_PROC_TIME     | BIGINT    | total_section_sort_proc_time - Total section sort processing time         |
| TOTAL_SECTION_SORTS              | BIGINT    | total_section_sorts - Total section sorts                                 |
| TOTAL_SORTS                      | BIGINT    | total_sorts - Total sorts   |
| POST_THRESHOLD_SORTS             | BIGINT    | post_threshold_sorts - Post threshold sorts                               |
| POST_SHRTHRESHOLD_SORTS          | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts                     |
| SORT_OVERFLOWS                   | BIGINT    | sort_overflows - Sort overflows   |
| TOTAL_COMPILE_TIME               | BIGINT    | total_compile_time - Total compile time                                   |
| TOTAL_COMPILE_PROC_TIME          | BIGINT    | total_compile_proc_time - Total compile processing time                   |
| TOTAL_COMPILATIONS               | BIGINT    | total_compilations - Total compilations                                   |
| TOTAL_IMPLICIT_COMPILE_TIME      | BIGINT    | total_implicit_compile_time - Total implicit compile time                 |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT    | total_implicit_compile_proc_time - Total implicit compile processing time |
| TOTAL_IMPLICIT_COMPILATIONS      | BIGINT    | total_implicit_compilations - Total implicit complications                |
| TOTAL_SECTION_TIME               | BIGINT    | total_section_time - Total section time                                   |
| TOTAL_SECTION_PROC_TIME          | BIGINT    | total_section_proc_time - Total section processing time                   |
| TOTAL_APP_SECTION_EXECUTIONS     | BIGINT    | total_app_section_executions - Total application section executions       |
| TOTAL_ACT_TIME                   | BIGINT    | total_act_time - Total activity time                                      |
| TOTAL_ACT_WAIT_TIME              | BIGINT    | total_act_wait_time - Total activity wait time                            |
| ACT_RQSTS_TOTAL                  | BIGINT    | act_rqsts_total - Total activity requests                                 |
| TOTAL_ROUTINE_TIME               | BIGINT    | total_routine_time - Total routine time                                   |
| TOTAL_ROUTINE_INVOCATIONS        | BIGINT    | total_routine_invocations - Total routine invocations                     |
| TOTAL_COMMIT_TIME                | BIGINT    | total_commit_time - Total commit time                                     |
| TOTAL_COMMIT_PROC_TIME           | BIGINT    | total_commit_proc_time - Total commits processing time                    |
| TOTAL_APP_COMMITS                | BIGINT    | total_app_commits - Total application commits                             |
| INT_COMMITS                      | BIGINT    | int_commits - Internal commits  |
| TOTAL_ROLLBACK_TIME              | BIGINT    | total_rollback_time - Total rollback time                                 |
| TOTAL_ROLLBACK_PROC_TIME         | BIGINT    | total_rollback_proc_time - Total rollback processing time                 |
| TOTAL_APP_ROLLBACKS              | BIGINT    | total_app_rollbacks - Total application rollbacks                         |
| INT_ROLLBACKS                    | BIGINT    | int_rollbacks - Internal rollbacks  |
| TOTAL_RUNSTATS_TIME              | BIGINT    | total_runstats_time - Total runtime statistics time                       |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                 | Data Type | Description   |
|-----------------------------|-----------|---|
| TOTAL_RUNSTATS_PROC_TIME    | BIGINT    | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS              | BIGINT    | total_runstats - Total runtime statistics                           |
| TOTAL_REORG_TIME            | BIGINT    | total_reorg_time - Total reorganization time                        |
| TOTAL_REORG_PROC_TIME       | BIGINT    | total_reorg_proc_time - Total reorganization processing time        |
| TOTAL_REORGS                | BIGINT    | total_reorgs - Total reorganizations                                |
| TOTAL_LOAD_TIME             | BIGINT    | total_load_time - Total load time                                   |
| TOTAL_LOAD_PROC_TIME        | BIGINT    | total_load_proc_time - Total load processing time                   |
| TOTAL_LOADS                 | BIGINT    | total_loads - Total loads   |
| CAT_CACHE_INSERTS           | BIGINT    | cat_cache_inserts - Catalog cache inserts                           |
| CAT_CACHE_LOOKUPS           | BIGINT    | cat_cache_lookups - Catalog cache lookups                           |
| PKG_CACHE_INSERTS           | BIGINT    | pkg_cache_inserts - Package cache inserts                           |
| PKG_CACHE_LOOKUPS           | BIGINT    | pkg_cache_lookups - Package cache lookups                           |
| THRESH_VIOLATIONS           | BIGINT    | thresh_violations - Number of threshold violations                  |
| NUM_LW_THRESH_EXCEEDED      | BIGINT    | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded    |
| FCM_TQ_RECV_WAIT_TIME       | BIGINT    | fcm_tq_recv_wait_time - FCM table queue received wait time          |
| FCM_MESSAGE_RECV_WAIT_TIME  | BIGINT    | fcm_message_recv_wait_time - FCM message received wait time         |
| FCM_TQ_SEND_WAIT_TIME       | BIGINT    | fcm_tq_send_wait_time - FCM table queue send wait time              |
| FCM_MESSAGE_SEND_WAIT_TIME  | BIGINT    | fcm_message_send_wait_time - FCM message send wait time             |
| AUDIT_FILE_WRITE_WAIT_TIME  | BIGINT    | audit_file_write_wait_time - Audit file write wait time             |
| AUDIT_FILE_WRITES_TOTAL     | BIGINT    | audit_file_writes_total - Total audit files written                 |
| AUDIT_SUBSYSTEM_WAIT_TIME   | BIGINT    | audit_subsystem_wait_time - Audit subsystem wait time               |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT    | audit_subsystem_waits_total - Total audit subsystem waits           |
| DIAGLOG_WRITE_WAIT_TIME     | BIGINT    | diaglog_write_wait_time - Diagnostic log file write wait time       |
| DIAGLOG_WRITES_TOTAL        | BIGINT    | diaglog_writes_total - Total diagnostic log file writes             |
| FCM_MESSAGE_SENDS_TOTAL     | BIGINT    | fcm_message_sends_total - Total FCM message sends                   |
| FCM_MESSAGE_RECVS_TOTAL     | BIGINT    | fcm_message_recvs_total - Total FCM message receives                |
| FCM_MESSAGE_SEND_VOLUME     | BIGINT    | fcm_message_send_volume - FCM message send volume                   |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                       | Data Type | Description   |
|-----------------------------------|-----------|---|
| FCM_MESSAGE_RECV_VOLUME           | BIGINT    | fcm_message_recv_volume - FCM message received volume                       |
| FCM_TQ_SENDS_TOTAL                | BIGINT    | fcm_tq_sends_total - FCM table queue send total                             |
| FCM_TQ_RECVS_TOTAL                | BIGINT    | fcm_tq_recvs_total - FCM table queue receives total                         |
| FCM_TQ_SEND_VOLUME                | BIGINT    | fcm_tq_send_volume - FCM table queue send volume                            |
| FCM_TQ_RECV_VOLUME                | BIGINT    | fcm_tq_recv_volume - FCM table queue received volume                        |
| TQ_TOT_SEND_SPILLS                | BIGINT    | tq_tot_send_spills - Total number of table queue buffers overflowed         |
| AUDIT_EVENTS_TOTAL                | BIGINT    | audit_events_total - Total audit events                                     |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT    | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME      | BIGINT    | total_routine_user_code_time - Total routine user code time                 |
| LOCK_WAITS_GLOBAL                 | BIGINT    | lock_waits_global - Lock waits global                                       |
| LOCK_WAIT_TIME_GLOBAL             | BIGINT    | lock_wait_time_global - Lock wait time global                               |
| LOCK_TIMEOUTS_GLOBAL              | BIGINT    | lock_timeouts_global - Lock timeouts global                                 |
| LOCK_ESCALS_MAXLOCKS              | BIGINT    | lock_escals_maxlocks - Number of maxlocks lock escalations                  |
| LOCK_ESCALS_LOCKLIST              | BIGINT    | lock_escals_locklist - Number of locklist lock escalations                  |
| LOCK_ESCALS_GLOBAL                | BIGINT    | lock_escals_global - Number of global lock escalations                      |
| RECLAIM_WAIT_TIME                 | BIGINT    | reclaim_wait_time - Reclaim wait time                                       |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME    | BIGINT    | spacemappage_reclaim_wait_time - Space map page reclaim wait time           |
| CF_WAITS                          | BIGINT    | cf_waits - Number of cluster caching facility DB2 pureScale server waits    |
| CF_WAIT_TIME                      | BIGINT    | cf_wait_time - cluster caching facility wait time                           |
| POOL_DATA_GBP_L_READS             | BIGINT    | pool_data_gbp_l_reads - Group buffer pool data logical reads                |
| POOL_DATA_GBP_P_READS             | BIGINT    | pool_data_gbp_p_reads - Group buffer pool data physical reads               |
| POOL_DATA_LBP_PAGES_FOUND         | BIGINT    | pool_data_lbp_pages_found - Local buffer pool found data pages              |
| POOL_DATA_GBP_INVALID_PAGES       | BIGINT    | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages          |
| POOL_INDEX_GBP_L_READS            | BIGINT    | pool_index_gbp_l_reads - Group buffer pool index logical reads              |
| POOL_INDEX_GBP_P_READS            | BIGINT    | pool_index_gbp_p_reads - Group buffer pool index physical reads             |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                       | Data Type | Description  |
|-----------------------------------|-----------|--|
| POOL_INDEX_LBP_PAGES_FOUND        | BIGINT    | pool_index_lbp_pages_found - Local buffer pool index pages found                 |
| POOL_INDEX_GBP_INVALID_PAGES      | BIGINT    | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages             |
| POOL_XDA_GBP_L_READS              | BIGINT    | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests          |
| POOL_XDA_GBP_P_READS              | BIGINT    | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests         |
| POOL_XDA_LBP_PAGES_FOUND          | BIGINT    | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found                |
| POOL_XDA_GBP_INVALID_PAGES        | BIGINT    | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages            |
| EVMON_WAIT_TIME                   | BIGINT    | evmon_wait_time - Event monitor wait time  |
| EVMON_WAITS_TOTAL                 | BIGINT    | evmon_waits_total - Event monitor total waits                                    |
| TOTAL_EXTENDED_LATCH_WAIT_TIME    | BIGINT    | total_extended_latch_wait_time - Total extended latch wait time                  |
| TOTAL_EXTENDED_LATCH_WAITS        | BIGINT    | total_extended_latch_waits - Total extended latch waits                          |
| TOTAL_STATS_FABRICATION_TIME      | BIGINT    | total_stats_fabrication_time - Total statistics fabrication time                 |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT    | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| TOTAL_STATS_FABRICATIONS          | BIGINT    | total_stats_fabrications - Total statistics fabrications                         |
| TOTAL_SYNC_RUNSTATS_TIME          | BIGINT    | total_sync_runstats_time - Total synchronous RUNSTATS time                       |
| TOTAL_SYNC_RUNSTATS_PROC_TIME     | BIGINT    | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time       |
| TOTAL_SYNC_RUNSTATS               | BIGINT    | total_sync_runstats - Total synchronous RUNSTATS activities                      |
| TOTAL_DISP_RUN_QUEUE_TIME         | BIGINT    | total_disp_run_queue_time - Total dispatcher run queue time                      |
| TOTAL_PEDS                        | BIGINT    | total_peds - Total partial early distincts                                       |
| DISABLED_PEDS                     | BIGINT    | “disabled_peds - Disabled partial early distincts monitor element” on page 802   |
| POST_THRESHOLD_PEDS               | BIGINT    | post_threshold_peds - Partial early distincts threshold                          |
| TOTAL_PEAS                        | BIGINT    | total_peas - Total partial early aggregations                                    |
| POST_THRESHOLD_PEAS               | BIGINT    | post_threshold_peas - Partial early aggregation threshold                        |
| TQ_SORT_HEAP_REQUESTS             | BIGINT    | tq_sort_heap_requests - Table queue sort heap requests                           |
| TQ_SORT_HEAP_REJECTIONS           | BIGINT    | tq_sort_heap_rejections - Table queue sort heap rejections                       |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                        | Data Type | Description  |
|------------------------------------|-----------|--|
| POOL_QUEUED_ASYNC_DATA_REQS        | BIGINT    | pool_queued_async_data_reqs - Data prefetch requests   |
| POOL_QUEUED_ASYNC_INDEX_REQS       | BIGINT    | pool_queued_async_index_reqs - Index prefetch requests   |
| POOL_QUEUED_ASYNC_XDA_REQS         | BIGINT    | pool_queued_async_xda_reqs - XDA prefetch requests   |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces           |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces         |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces        |
| POOL_QUEUED_ASYNC_OTHER_REQS       | BIGINT    | pool_queued_async_other_reqs - Non-prefetch requests   |
| POOL_QUEUED_ASYNC_DATA_PAGES       | BIGINT    | pool_queued_async_data_pages - Data pages prefetch requests                                    |
| POOL_QUEUED_ASYNC_INDEX_PAGES      | BIGINT    | pool_queued_async_index_pages - Index pages prefetch requests                                  |
| POOL_QUEUED_ASYNC_XDA_PAGES        | BIGINT    | pool_queued_async_xda_pages - XDA pages prefetch requests                                      |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES  | BIGINT    | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces    |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT    | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces  |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES   | BIGINT    | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS        | BIGINT    | pool_failed_async_data_reqs - Failed data prefetch requests                                    |
| POOL_FAILED_ASYNC_INDEX_REQS       | BIGINT    | pool_failed_async_index_reqs - Failed index prefetch requests                                  |
| POOL_FAILED_ASYNC_XDA_REQS         | BIGINT    | pool_failed_async_xda_reqs - Failed XDA prefetch requests                                      |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces    |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces  |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces      |
| POOL_FAILED_ASYNC_OTHER_REQS       | BIGINT    | pool_failed_async_other_reqs - Failed non-prefetch requests                                    |

Table 38. Information returned for a unit of work event monitor: Table name: UOW\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the UOW\_EVENT table (continued)

| Column Name                             | Data Type | Description   |
|---|-----------|---|
| PREFETCH_WAIT_TIME                      | BIGINT    | prefetch_wait_time - Time waited for prefetch   |
| PREFETCH_WAITS                          | BIGINT    | prefetch_waits - Prefetcher wait count  |
| APP_ACT_COMPLETED_TOTAL                 | BIGINT    | app_act_completed_total - Total successful external coordinator activities  |
| APP_ACT_ABORTED_TOTAL                   | BIGINT    | app_act_aborted_total - Total failed external coordinator activities  |
| APP_ACT_REJECTED_TOTAL                  | BIGINT    | app_act_rejected_total - Total rejected external coordinator activities   |
| TOTAL_CONNECT_REQUEST_TIME              | BIGINT    | total_connect_request_time - Total connection or switch user request time   |
| TOTAL_CONNECT_REQUEST_PROC_TIME         | BIGINT    | total_connect_request_proc_time - Total connection or switch user request processing time   |
| TOTAL_CONNECT_REQUESTS                  | BIGINT    | total_connect_requests - Connection or switch user requests   |
| TOTAL_CONNECT_AUTHENTICATION_TIME       | BIGINT    | total_connect_authentication_time - Total connection or switch user authentication request time   |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME  | BIGINT    | total_connect_authentication_proc_time - Total connection authentication processing time  |
| TOTAL_CONNECT_AUTHENTICATIONS           | BIGINT    | total_connect_authentications - Connections or switch user authentications performed  |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP  | BIGINT    | "pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element" on page 1041   |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | "pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element" on page 1073 |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP   | BIGINT    | "pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element" on page 1130     |
| COMM_EXIT_WAIT_TIME                     | BIGINT    | comm_exit_wait_time - Communication buffer exit wait time monitor element   |
| COMM_EXIT_WAITS                         | BIGINT    | comm_exit_waits - Communication buffer exit number of waits monitor element   |

**Information written to XML by EVMON\_FORMAT\_UE\_TO\_XML for a unit of work event monitor:**

Information written for a unit of work event monitor from the EVMON\_FORMAT\_UE\_TO\_XML table function. This is also documented in the sql/lib/misc/DB2EvmonUOW.xsd file.

## db2\_uow\_event

The main schema that describes a unit of work event.

**Element content:** ( "completion\_status" on page 195, "start\_time" on page 195, "stop\_time" on page 195, "connection\_time" on page 196, "application\_name" on page 196, "application\_handle" on page 196, "application\_id" on page 196, "uow\_id" on page 196, "workload\_occurrence" on page 196, "coord\_member" on page 196, "member\_activation\_time" on page 197, "workload\_name" on page 197, "workload\_id" on page 197, "service\_superclass\_name" on page 197 {zero or one times (?)}, "service\_subclass\_name" on page 197 {zero or one times (?)}, "service\_class\_id" on page 197 {zero or one times (?)}, "session\_authid" on page 198 {zero or one times (?)}, "system\_authid" on page 198, "client\_pid" on page 198, "client\_product\_id" on page 198, "client\_platform" on page 198, "client\_protocol" on page 198 {zero or one times (?)}, "client\_userid" on page 199 {zero or one times (?)}, "client\_wrkstnname" on page 199 {zero or one times (?)}, "client\_applname" on page 199 {zero or one times (?)}, "client\_acctng" on page 199 {zero or one times (?)}, "local\_transaction\_id" on page 199, "global\_transaction\_id" on page 199, "system\_metrics" on page 199, "client\_hostname" on page 199, "client\_port\_number" on page 200, "uow\_log\_space\_used" on page 200, "package\_list" on page 200, "executable\_list" on page 200, "intra\_parallel\_state" on page 201, ANY content ( skip ) {zero or more (\*)} )

### Attributes:

| QName                            | Type        | Fixed | Default | Use      | Annotation |
|----------------------------------|-------------|-------|---------|----------|------------|
| id                               | xs:long     |       |         | required |            |
| type                             |             |       |         | required |            |
| timestamp                        | xs:dateTime |       |         | required |            |
| member                           |             |       |         | required |            |
| release                          | xs:long     |       |         | required |            |
| mon_interval_id                  | xs:long     |       |         | required |            |
| ANY attribute from ANY namespace |             |       |         |          |            |

## package\_id

See monitor element "package\_id - Package identifier monitor element" on page 1005 for more details.

**Contained by:** "package\_entry" on page 190

### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### package\_elapsed\_time

See monitor element “package\_elapsed\_time - Package elapsed time monitor element” on page 1005 for more details.

**Contained by:** “package\_entry”

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### invocation\_id

See monitor element “invocation\_id - Invocation ID monitor element” on page 886 for more details.

**Contained by:** “package\_entry”

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

### routine\_id

See monitor element “routine\_id - Routine ID monitor element” on page 1187 for more details.

**Contained by:** “package\_entry”

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

### nesting\_level

See monitor element “nesting\_level - Nesting level monitor element” on page 967 for more details.

**Contained by:** “package\_entry”

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

### package\_entry

**Contained by:** “package\_list\_entries” on page 191

**Element content:** ( “package\_id” on page 189, “package\_elapsed\_time” on page 190, “invocation\_id” on page 190, “routine\_id” on page 190, “nesting\_level” on page 190, ANY content ( skip ) {zero or more (\*)} )

**Attributes:**

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

**package\_list\_size**

**Contained by:** “package\_list” on page 200

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

**package\_list\_exceeded**

See monitor element “package\_list\_exceeded - Package list exceeded monitor element” on page 1006 for more details.

**Contained by:** “package\_list” on page 200

**package\_list\_entries**

**Contained by:** “package\_list” on page 200

**Element content:** ( “package\_entry” on page 190 {zero or more (\*)} )

**Attributes:**

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

**executable\_id**

See monitor element “executable\_id - Executable ID monitor element” on page 818 for more details.

**Contained by:** “executable\_entry” on page 194

**num\_executions**

See monitor element “num\_executions - Statement executions monitor element” on page 973 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**rows\_read**

See monitor element “rows\_read - Rows read monitor element” on page 1191 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_cpu\_time**

See monitor element “total\_cpu\_time - Total CPU time monitor element” on page 1330 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_act\_time**

See monitor element “total\_act\_time - Total activity time monitor element” on page 1311 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_act\_wait\_time**

See monitor element “total\_act\_wait\_time - Total activity wait time monitor element” on page 1312 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_wait\_time

See monitor element “lock\_wait\_time - Time waited on locks monitor element” on page 924 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_waits

See monitor element “lock\_waits - Lock waits monitor element” on page 928 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### total\_sorts

See monitor element “total\_sorts - Total sorts monitor element” on page 1375 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### post\_threshold\_sorts

See monitor element “post\_threshold\_sorts - Post threshold sorts monitor element” on page 1150 for more details.

**Contained by:** “executable\_entry” on page 194

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### post\_shrthreshold\_sorts

See monitor element “post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144 for more details.

**Contained by:** “executable\_entry”

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### sort\_overflows

See monitor element “sort\_overflows - Sort overflows monitor element” on page 1224 for more details.

**Contained by:** “executable\_entry”

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### executable\_entry

**Contained by:** “executable\_list\_entries” on page 195

**Element content:** ( “executable\_id” on page 191, “num\_executions” on page 191, “rows\_read” on page 192, “total\_cpu\_time” on page 192, “total\_act\_time” on page 192, “total\_act\_wait\_time” on page 192, “lock\_wait\_time” on page 193, “lock\_waits” on page 193, “total\_sorts” on page 193, “post\_threshold\_sorts” on page 193, “post\_shrthreshold\_sorts”, “sort\_overflows”, ANY content ( skip ) {zero or more (\*)} )

**Attributes:**

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### executable\_list\_size

**Contained by:** “executable\_list” on page 200

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

### executable\_list\_truncated

**Contained by:** “executable\_list” on page 200

### executable\_list\_entries

**Contained by:** “executable\_list” on page 200

**Element content:** ( “executable\_entry” on page 194 {zero or more (\*)} )

#### Attributes:

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### completion\_status

The completion status of the unit of work. Possible values are: UNKNOWN, COMMIT, ROLLBACK, GLOBAL\_COMMIT, GLOBAL\_ROLLBACK, XA\_END, XA\_PREPARE

**Contained by:** “db2\_uow\_event” on page 189

### start\_time

The start time of the unit of work. See monitor element “uow\_start\_time - Unit of work start timestamp monitor element” on page 1404 for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### stop\_time

The stop time of the unit of work. See monitor element “uow\_stop\_time - Unit of work stop timestamp monitor element” on page 1405 for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

**connection\_time**

The time the application connected to the database member. See monitor element “conn\_time - Time of database connection monitor element” on page 736 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

**application\_name**

The name of the application running at the client, as known to the database. See monitor element “appl\_name - Application name monitor element” on page 670 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**application\_handle**

A system-wide unique ID for the application. See monitor element “agent\_id - Application handle (agent ID) monitor element” on page 650 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**application\_id**

This identifier is generated when the application connects to the database at the database manager. See monitor element “appl\_id - Application ID monitor element” on page 666 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**uow\_id**

The unit of work ID to which this activity record applies. See monitor element “uow\_id - Unit of work ID monitor element” on page 1401 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**workload\_occurrence\_id**

The workload occurrence ID to which this activity record applies. See monitor element “workload\_occurrence\_id - Workload occurrence identifier monitor element” on page 1428 for more details.

**Contained by:** “db2\_uow\_event” on page 189

**coord\_member**

**Contained by:** “db2\_uow\_event” on page 189

### **member\_activation\_time**

The time this database member was activated. See monitor element “db\_conn\_time - Database activation timestamp monitor element” on page 772 for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### **Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### **workload\_name**

The name of the workload under which the unit of work completed. See monitor element “workload\_name - Workload name monitor element” on page 1427 for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **workload\_id**

The workload ID of the workload under which the unit of work completed. See monitor element “workload\_id - Workload ID monitor element” on page 1426 for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **service\_superclass\_name**

The name of the service super class under which the unit of work completed. See monitor element “service\_superclass\_name - Service superclass name monitor element” on page 1209 for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **service\_subclass\_name**

The name of the service sub class under which the unit of work completed. See monitor element “service\_subclass\_name - Service subclass name monitor element” on page 1208 for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **service\_class\_id**

The service class ID of the service class under which the unit of work completed. See monitor element “service\_class\_id - Service class ID monitor element” on page 1207 for more details.

**Contained by:** “db2\_uow\_event” on page 189

### session\_authid

The session authorization ID of the user who invoked the application that is being monitored. See monitor element “session\_auth\_id - Session authorization ID monitor element” on page 1211for more details.

**Contained by:** “db2\_uow\_event” on page 189

### system\_authid

The system authorization ID of the user who invoked the application that is being monitored. See monitor element “system\_auth\_id - System authorization identifier monitor element” on page 1267for more details.

**Contained by:** “db2\_uow\_event” on page 189

### client\_pid

The process ID reported by the client. See monitor element “client\_pid - Client process ID monitor element” on page 713for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### client\_product\_id

The product ID of the client. See monitor element “client\_prdid - Client product and version ID monitor element” on page 715for more details.

**Contained by:** “db2\_uow\_event” on page 189

### client\_platform

The platform of the client. See monitor element “client\_platform - Client operating platform monitor element” on page 714for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Attributes:

| QName | Type     | Fixed | Default | Use      | Annotation |
|-------|----------|-------|---------|----------|------------|
| id    | xs:short |       |         | optional |            |

### client\_protocol

The product ID of the client. See monitor element “client\_protocol - Client communication protocol monitor element” on page 716for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **client\_userid**

The client user ID generated by a transaction manager and provided to the server. See monitor element “client\_userid - Client user ID monitor element” on page 717for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **client\_wrkstname**

Identifies the client system or workstation, if the sqleseti API was issued in this connection. See monitor element “client\_wrkstname - Client workstation name monitor element” on page 717for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **client\_applname**

Identifies the server transaction program performing the transaction, if the sqleseti API was issued in this connection. See monitor element “client\_applname - Client application name monitor element” on page 710for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **client\_acctng**

The data passed to the target database for logging and diagnostic purposes, if the sqleseti API was issued in this connection. See monitor element “client\_acctng - Client accounting string monitor element” on page 709for more details.

**Contained by:** “db2\_uow\_event” on page 189

### **local\_transaction\_id**

The local transaction id for the unit of work.

**Contained by:** “db2\_uow\_event” on page 189

### **global\_transaction\_id**

The global transaction id for the unit of work.

**Contained by:** “db2\_uow\_event” on page 189

### **system\_metrics**

The metrics for the unit of work.

**Contained by:** “db2\_uow\_event” on page 189

### **client\_hostname**

The hostname of the client. See monitor element “client\_hostname - Client hostname monitor element” on page 711for more details.

**Contained by:** “db2\_uow\_event” on page 189

### client\_port\_number

The port number of the client. See monitor element “client\_port\_number - Client port number monitor element” on page 715 for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Element content:

| Type   | Facet |
|--------|-------|
| xs:int |       |

### uow\_log\_space\_used

The amount of log space used during the unit of work. See monitor element “uow\_log\_space\_used - Unit of work log space used monitor element” on page 1403 for more details.

**Contained by:** “db2\_uow\_event” on page 189

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### package\_list

The package list for the unit of work.

**Contained by:** “db2\_uow\_event” on page 189

**Element content:** ( “package\_list\_size” on page 191, “package\_list\_exceeded” on page 191, “package\_list\_entries” on page 191, ANY content ( skip ) {zero or more (\*)} )

#### Attributes:

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### executable\_list

The executable list for the unit of work.

**Contained by:** “db2\_uow\_event” on page 189

**Element content:** ( “executable\_list\_size” on page 194, “executable\_list\_truncated” on page 195, “executable\_list\_entries” on page 195, ANY content ( skip ) {zero or more (\*)} )

#### Attributes:

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### **intra\_parallel\_state**

The current intrapartition parallelism state of the unit of work. Possible values are YES and NO.

**Contained by:** “db2\_uow\_event” on page 189

### **Unit of work event monitor package listing information:**

The unit of work event monitor can collect a listing of packages used within a unit of work. This information can be used to determine which stored procedures within an application might be taking more time than expected to run.

Starting with DB2 Version 9.7 Fix Pack 1, you can have information about the packages used within units of work included in the data the event monitor collects. Depending on the output option you choose for the unit of work event monitor, this information is written to the unformatted event table or to the UOW\_PACKAGE\_LIST\_ *evmon-name* table (where *evmon-name* is the name assigned to the event monitor) when the unit of work ends along with the rest of the information associated with the event.

There are two ways to control the capture of this information:

- The PACKAGE LIST option for the COLLECT UNIT OF WORK DATA clause of the CREATE or ALTER WORKLOAD statements controls the collection of this information for *specific* workloads. If this option is specified, information for the units of work that are executed under the workload identified in the CREATE or ALTER WORKLOAD statements, including package list information, is sent to any active unit of work event monitors.
- The **mon\_uow\_pkglist** configuration parameter can be set to ON so that package list information for *all* units of work executed on the data server is sent to any active unit of work event monitors.

**Note:** **mon\_uow\_data** must also be set to BASE for package list information to be collected.

The following data is collected for the package listing:

#### **Package ID (“package\_id - Package identifier monitor element” on page 1005)**

A unique ID that identifies a package.

#### **Nesting level (“nesting\_level - Nesting level monitor element” on page 967)**

The level of nesting or recursion in effect when the statement was being run. Each level of nesting corresponds to nested or recursive invocation of a stored procedure or user-defined function (UDF).

#### **Routine ID (“routine\_id - Routine ID monitor element” on page 1187)**

A unique routine identifier. It returns zero if the activity is not part of a routine.

**Invocation ID (“`invocation_id` - Invocation ID monitor element” on page 886)**

An identifier that distinguishes one invocation of a routine from others at the same nesting level within a unit of work. It is unique within a unit of work for a specific nesting level.

**Package elapsed time (“`package_elapsed_time` - Package elapsed time monitor element” on page 1005)**

The elapsed time spent executing sections within the package.

As the list of information collected for the package listing suggests, information is captured not only for each package but also for each invocation of a routine within a package.

Elapsed time is also tracked. The time calculated for a given invocation starts from the first execution of a section within a package until the database manager switches to another package. See “Examples” on page 205 to see more about how elapsed time is tracked.

**How package lists are written to unformatted event tables**

When you enable the collection of package list information, the unit of work event monitor writes two records to the unformatted event (UE) table for each unit of work. The first record contains the basic unit of work event monitor data. The next record contains the package listing information.

Package list information is stored in the UE table in a BLOB column. A list with 32 entries can be stored as an inline BLOB when the page size for the table space is 4k (the default). The number of entries that can be written to the package list is controlled by the `mon_pkglist_sz` configuration parameter. The default for this parameter is 32, which means that up to 32 entries can be included in the package listing. If you want to increase the number of entries that can be included in the package list, ensure that the UE table used to store the event monitor output is created in a table space with a larger page size. Assume that every increase of 32 in the size of the package list requires an increase of 4k in the page size of the table space. So, for example, if you want to have up to 64 entries in the package list, ensure that the page size for the table space is at least 8k. If you increase `mon_pkglist_sz` without increasing the page size of the table space, the package list is still created, however the BLOB is not stored inline in the table, which might affect performance.

**Note:** You can use the `ADMIN_IS_INLINED` administrative function to determine whether the BLOB that contains the package list information is stored inline.

**How package lists are written to regular tables**

When you use regular tables for event monitor output, package list information is captured as part of the “`uow_package_list` logical data group” on page 87. As each unit of work completes, one or more rows are added to the `UOW_PACKAGE_LIST_evmon-name` table, with one column for each monitor element in the logical data group. The number of rows added to the table depends on how many packages ran as part of the unit of work. However, the upper limit to the number of rows that can be added to this table is controlled by the `mon_pkglist_sz` configuration parameter. The default for this parameter is 32, which means that up to 32 entries can be included in the package listing. If you want to increase the number of entries that can be included in the package list, increase `mon_pkglist_sz`.

## Package listing output

As stated earlier, when the event monitor writes to a UE table, the unit of work event monitor writes two records to the UE table when collecting package information. Each of the interfaces for displaying the data in a UE table provides a mechanism for viewing the information contained in the two UE table records. For example, the **db2evmonfmt** tool combines the information in each record into a single report. If you use the `EVMON_FORMAT_UE_TO_TABLES` procedure, it produces relational tables that you can join; the table `UOW_PACKAGE_LIST` contains the package list information. `EVMON_FORMAT_UE_TO_XML` produces a single XML document that contains the information from both records. For more information, see “Accessing event data that is captured by a unit of work event monitor” on page 165.

When the event monitor writes to relational tables directly, the package list information is written to the table `UOW_PACKAGE_LIST_<evmon-name>`.

**Note:** In a partitioned database environment, the package list is only reported in the unit of work event generated by the *coordinator agent* and reflects the time spent in each package by that agent specifically; it does not reflect time spent in those packages by any other agent at any other partition.

Figure 5 on page 204 shows the information produced by the unit of work event monitor, as formatted by the **db2evmonfmt** tool.

```

-----
Event ID          : 12
Event Type       : UOW
Event Timestamp  : 2009-12-08-14.44.39.162707
Member          : 0
Release         : 9070200
-----

Database Level Details
-----
Database Member Activation Time : 2009-12-08-14.41.55.089416
Coordinator Member             : 0

Connection Level Details
-----
Application ID                : *LOCAL.gstager.091208194155
Application Handle            : 21
Application Name              : db2bp
Session Authorization ID     :
System Authorization ID      :
Connection Timestamp         : 2009-12-08-14.41.55.089416
Client Process ID            : 13043
Client Platform              : LINUX8664
Client Product ID            : SQL09072
Client Protocol               : LOCAL
Client Hostname              : HOSTX
Client Port Number           : 0

UOW Level Details
-----
Start Time                  : 2009-12-08-14.44.39.160651
Stop Time                  : 2009-12-08-14.44.39.162707
Completion Status          : COMMIT
UOW ID                     : 12
Workload Occurrence ID    : 1
Workload Name              : SYSDEFAULTUSERWORKLOAD
Workload ID                : 1
Service Superclass Name    : SYSDEFAULTUSERCLASS
Service Subclass Name      : SYSDEFAULTSUBCLASS
Service Class ID           : 13
Client Userid              :
Client Workstation Name    :
Client Application Name    :
Client Accounting String   :
Local Transaction ID       : 0000000000000013B
Global Transaction ID      : 0000000000000000000000000000000000000000
Log Space Used             : 124

UOW Metrics
-----
TOTAL_CPU_TIME             : 1591
TOTAL_WAIT_TIME            : 8363
ACT_ABORTED_TOTAL         : 0
ACT_COMPLETED_TOTAL       : 1
ACT_REJECTED_TOTAL        : 0
AGENT_WAIT_TIME           : 87
AGENT_WAITS_TOTAL         : 1
APP_RQSTS_COMPLETED_TOTAL : 1
.
.
.

Package List
-----
Package List Size          : 2
Package List Exceeded     : no

PACKAGE_ID      NESTING_LEVEL ROUTINE_ID  INVOCATION_ID  PACKAGE_ELAPSED_TIME
-----
240             0             0             0             0
330             1             66539         1             1

```

**Note:** Some of the metrics in the "UOW Metrics" section have been excluded.

*Figure 5. Sample output from the unit of work event monitor, with package listing information*

The number of packages that appear in the package list for a given unit of work is reflected in the **package\_list\_count** monitor element ("Package List Size" in the preceding report), which is included with the base unit of work event monitor data. If the number of packages used with the unit of work exceeds the value

specified in the `mon_pkglist_sz` configuration parameter, the additional packages are not included in the package listing. However, the `package_list_exceeded` monitor element indicates whether there were more packages than would fit into the package list. This monitor element is returned along with the base information for the unit of work event monitor (“Package List Exceeded” in Figure 5 on page 204). If the value for this monitor element is YES, you can increase the value for `mon_pkglist_sz` to have a larger number of packages included in the package list.

### Examples

Each of the examples that follow show the information returned for the package listing as it would be displayed by the `db2evmonfmt` tool.

*Example 1: An application that executes one or more sections in a single package*

In this example, one package with a package ID of 300 was run for this unit of work.

| PACKAGE_ID | NESTING_LEVEL | ROUTINE_ID | INVOCATION_ID | ELAPSED_TIME |
|------------|---------------|------------|---------------|--------------|
| 300        | 0             | 0          | 0             | 100          |

In this case, there is one entry on the package list, which reflects the execution of one or more sections in the package. All sections executed from the same package are considered to be part of the same package invocation.

*Example 2: An application calls a stored procedure in a package*

In this example, the package with a package ID of 300 calls a stored procedure with an ID of 806. Three sections are executed within the stored procedure.

| PACKAGE_ID | NESTING_LEVEL | ROUTINE_ID | INVOCATION_ID | ELAPSED_TIME |
|------------|---------------|------------|---------------|--------------|
| 300        | 0             | 0          | 0             | 21           |
| 300        | 1             | 806        | 1             | 100          |

This output shows two entries in the list. One entry is for the call to the stored procedure, and one for the execution of the three sections within the stored procedure. The `NESTING_LEVEL` for the second entry in the list reflects the fact that the stored procedure was called from another package.

*Example 3: An application executes sections in two different packages*

In this example, an application executes sections from one package, then another package, and then back to the first package. No stored procedures are called. The pseudocode that follows is a representation of this unit of work:

```
Application
EXEC PACKAGEA
EXEC PACKAGEB
EXEC PACKAGEA
```

Assume also that the invocation of `PACKAGEA` requires 100 ms, the invocation of `PACKAGEB` requires 25 ms, and that the invocation of `PACKAGEC` requires 460 ms. The following output shows what the package listing would look like:

| PACKAGE_ID | NESTING_LEVEL | ROUTINE_ID | INVOCATION_ID | ELAPSED_TIME |
|------------|---------------|------------|---------------|--------------|
| 300        | 0             | 0          | 0             | 560          |
| 301        | 0             | 0          | 0             | 25           |

In this case, there are two entries in the list. Package A, with PACKAGE\_ID 300 had sections that ran for 560 ms in total. Package B ran for 25 ms. Package A is represented by a single line because each invocation has the same INVOCATION\_ID and NESTING\_LEVEL. INVOCATION\_ID and NESTING\_LEVEL remain at 0, because no stored procedures were called in either package.

*Example 4: An application executes sections and stored procedures in multiple packages*

In this example, there are 3 packages with IDs 100, 101, and 102. The application is in package 100. There are two stored procedures with IDs 201 and 202. The first stored procedure (SP1) is in package 101, and the second (SP2) is in package 102. The pseudocode that follows is a representation of this unit of work:

```
Application
CALL SP1 a
    INSERT INTO T1 VALUES(7) b
    CALL SP2 c
        INSERT INTO T2 VALUES(8)
    CALL SP2 d
        INSERT INTO T2 VALUES(8)
```

The package listing for this unit of work would be as follows:

| PACKAGE_ID | NESTING_LEVEL | ROUTINE_ID | INVOCATION_ID | ELAPSED_TIME |
|------------|---------------|------------|---------------|--------------|
| 100        | 0             | 0          | 0             | 21           |
| 101        | 1 <b>1</b>    | 201        | 1             | 40           |
| 102        | 2 <b>2</b>    | 202        | 1 <b>3</b>    | 35           |
| 102        | 2             | 202        | 2 <b>3</b>    | 35           |

In the preceding output, there are four entries:

- The first corresponds to the execution of the call to SP1 within the first package, line **a** in the pseudocode that represents the unit of work.
- The second corresponds to the execution of the sections within the stored procedure with ID 201 in package 101. These sections include lines **b**, **c**, and **d**. The nesting level increases to 1, as shown by **1**.
- The third entry represents the execution of the first of the INSERT INTO T2 statements in SP2, as called from SP1. The nesting level increases again (**2**).
- The fourth entry in the list represents the execution of the second of the INSERT INTO T2 statements in SP2. The nesting level remains the same, because like the previous call to SP2, this stored procedure is called from SP1. However, because these two statements occur within separate invocations of the stored procedure, they each have separate invocation IDs (**3**). Thus, there are two separate entries in the package listing.

#### Executable list information:

When collecting unit of work information, you have the option to also collect a list of the executable IDs of statements that run as part of each unit of work.

The information about the executable IDs is written to an unformatted event (UE) table or to a regular table. There are two ways to capture this information:

- Use the EXECUTABLE LIST option for the COLLECT UNIT OF WORK DATA clause of the CREATE WORKLOAD or ALTER WORKLOAD statement to collect information for specific workloads. Information for the units of work that are executed under the workload that you identify in the statement, including executable IDs, is sent to the active unit of work (UOW) event monitors.

- Use configuration parameters to have information about all units of work that are executed on the data server, including executable information, sent to active unit of work event monitors. To collect executable ID information, set the **mon\_uow\_data** configuration parameter to BASE, and set the **mon\_uow\_exclist** configuration parameter to ON.

The following data is collected for the executable listing:

#### Unit of Work Level

##### **executable\_list\_size**

The number of entries that are present within the executable ID listing for a particular unit of work.

##### **executable\_list\_truncated**

A YES or NO value indicating whether the list is truncated. The list can be truncated if there is insufficient memory available to store the entire executable list during processing.

#### Executable ID List

**executable\_id** (“executable\_id - Executable ID monitor element” on page 818) An opaque binary token generated on the data server that uniquely identifies the SQL statement section that was executed.

**num\_executions** (“num\_executions - Statement executions monitor element” on page 973)

The number of times that an SQL statement has been executed.

**rows\_read** (“rows\_read - Rows read monitor element” on page 1191)

The number of rows read from the table.

**total\_cpu\_time** (“total\_cpu\_time - Total CPU time monitor element” on page 1330)

The total amount of CPU time used while in the DB2 product. Represents total of both user and system CPU time. This value is given in microseconds.

**total\_act\_time** (“total\_act\_time - Total activity time monitor element” on page 1311)

The total amount of time spent executing activities. This value is given in milliseconds.

**total\_act\_wait\_time** (“total\_act\_wait\_time - Total activity wait time monitor element” on page 1312)

Total time spent waiting within the DB2 database server, while processing an activity. The value is given in milliseconds.

**lock\_wait\_time** (“lock\_wait\_time - Time waited on locks monitor element” on page 924)

The total elapsed time spent waiting for locks. The value is given in milliseconds.

**lock\_waits** (“lock\_waits - Lock waits monitor element” on page 928)

The total number of times that applications or connections waited for locks.

**total\_sorts** (“total\_sorts - Total sorts monitor element” on page 1375)

The total number of sorts that have been executed.

**post\_threshold\_sorts (“post\_threshold\_sorts - Post threshold sorts monitor element” on page 1150)**

The number of sorts that have requested heaps after the sort heap threshold has been exceeded.

**post\_shrthreshold\_sorts (“post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144)**

The total number of sorts that were throttled back by the sort memory throttling algorithm. A throttled sort is a sort that was granted less memory than requested by the sort memory manager

**sort\_overflows (“sort\_overflows - Sort overflows monitor element” on page 1224)**

The total number of sorts that ran out of sort heap and may have required disk space for temporary storage.

### **How execution lists are written to UE tables**

At least two separate records can be written to the UE table when base data and executable ID list data is collected for an UOW event monitor. The first record contains information about the UOW event that contains the base UOW data. The second record is the UOW\_EXEC\_LIST event that contains the executable ID listing data. This second record might consist of multiple records because of the potentially large number of unique executable IDs for a single UOW. These records are written as separate rows to the UE table to ensure that each event is contained within the inlined LOB space available. You can use the interfaces for formatting the UE table to merge information for these events. When the executable ID list is not collected, an associated record is not created; the table does not contain any rows.

### **How execution lists are written to regular tables**

When you use regular tables for event monitor output, executable list information is captured as part of the “uow\_executable\_list logical data group” on page 81. As each unit of work is completed, one or more rows are added to the table UOW\_EXECUTABLE\_LIST\_ *evmon\_name*, with one column for each monitor element in the logical data group. The number of rows that are added to the table depends on how many unique executable IDs ran as part of the unit of work.

### **Executable listing output**

When the event monitor writes to a UE table, the unit of work event monitor writes two records to the UE table when collecting execution information. Each of the interfaces for displaying the data in a UE table provides a mechanism for viewing the information contained in the two UE table records. The **db2evmonfmt** tool combines the information in each record into a single report. The EVMON\_FORMAT\_UE\_TO\_TABLES procedure produces relational tables that you can join; the table UOW\_EXECUTABLE\_LIST contains the executable list information. The EVMON\_FORMAT\_UE\_TO\_XML table function produces a single XML document that contains the information from both records. For more information, see “Accessing event data that is captured by a unit of work event monitor” on page 165.

When the event monitor writes directly to relational tables, the executable list information is written to the table UOW\_EXECUTABLE\_LIST\_ *evmon\_name*.

In a partitioned database environment, the executable ID list is generated per member, including per coordinator agent member and data member. In a DB2 pureScale environment, the list is generated from the coordinator member, which is similar to the situation in non-partitioned configurations.

### Examples

The following sample information was collected for an application that executes five different SQL statement sections within a UOW. This output provides a logical view with sample columns; the actual output depends on the tool or query that you run.

| EXECUTABLE_ID                          | NUM_EXECUTIONS | ROWS_READ | TOTAL_CPU_TIME |
|--|----------------|-----------|----------------|
| x'01007A00000020020081126171554951791' | 1              | 23456     | 76888          |
| x'01007900000020020081126171533551120' | 55             | 345       | 768            |
| x'01007C00000020020081126171720728997' | 234            | 67        | 232            |
| x'01007B00000020020081126171657272914' | 3456           | 347       | 1223           |
| x'01007D00000020020081126172409987719' | 22242          | 2244      | 432444         |

In this example, there are five entries in the executable ID list to correspond to the five different sections that were executed. The five sections were executed a different number of times, as illustrated by the NUM\_EXECUTIONS column, but only one entry is provided for each unique section. The first row might indicate a problematic activity statement because it consumed excessive CPU time in just one execution.

### Collecting unit of work event data and generating reports

You can use the unit of work event monitor to collect data about transactions that you can use for chargeback purposes. The collected transaction event data is in an unreadable form in an unformatted event table. You can use this data to create a readable text report.

### Before you begin

To collect unit of work event monitor data, you must have SYSADM or SYSCTRL authority.

### About this task

This task provides instructions for collecting unit of work event data for a particular workload.

Package listing and execution list information is also collected if you set both the **mon\_uow\_pkglist** and **mon\_uow\_execlist** configuration parameters to ON. Alternatively, you can collect package listing and execution list information for a workload, regardless of the settings of the **mon\_uow\_pkglist** and **mon\_uow\_execlist** configuration parameters, by altering the ALTER WORKLOAD statement as follows:

- For package listing information, replace the BASE option with the BASE INCLUDE PACKAGE LIST option.
- For execution list information, replace the BASE option with the BASE INCLUDE EXECUTABLE LIST option.
- For package listing and execution list information, replace the BASE option with the BASE INCLUDE PACKAGE LIST, EXECUTABLE LIST option.

The unit of work event monitor collects information that identifies application transactions and the corresponding CPU usage. Examples of information that the unit of work event monitor collects for a transaction event are as follows:

- Total CPU usage time (TOTAL\_CPU\_TIME monitor element)
- Application handle (APPLICATION\_HANDLE monitor element)

#### Restrictions

Input data values are not viewable if you do not have SYSADM or SYSCTRL authority.

### Procedure

To collect detailed information regarding unit of work events:

1. Create a unit of work event monitor called UOWEVMON by issuing the CREATE EVENT MONITOR FOR UNIT OF WORK statement, as shown in the following example:

```
CREATE EVENT MONITOR UOWEVMON FOR UNIT OF WORK
WRITE TO UNFORMATTED EVENT TABLE
```

2. Activate the UOWEVMON unit of work event monitor by issuing the following statement:

```
SET EVENT MONITOR UOWEVMON STATE 1
```

3. Enable unit of work event data collection at the workload level by issuing the ALTER WORKLOAD statement with statement history. For example, to collect unit of work data for the FINANCE and PAYROLL applications, issue the following statements:

```
ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA BASE
ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA BASE
```

4. To collect unit of work transaction events, rerun the workload.
5. Connect to the database.
6. Produce a flat-text report that is based on the event data that is collected in the unformatted event table by using the XML parser tool, **db2evmonfmt**, as shown in the following example:

```
java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
```

7. Analyze the report to determine how much CPU time applications are using so that appropriate charges can be billed.
8. If you want to turn off unit of work data collection for both the FINANCE and PAYROLL applications, issue the following statements:

```
ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA NONE
ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA NONE
```

### Example

The following example of a report was obtained by using the **db2evmonfmt** tool to convert the data that the unit of work event monitor collected in the unformatted event table:

```
-----
Event ID           : 1
Event Type        : UOW
Event Timestamp    : 2008-10-31-13.29.04.130849
Member of detection : 0
-----
```

Database Level Details

```

-----
Member Activation Time      : 2008-10-31T13:28:48.538973
Coordinator Member         : 0

Connection Level Details
-----
Application ID              : *LOCAL.gstager.081031172848
Application Handle          : 20
Application Name            : db2bp
Session Authorization ID    : GSTAGER
System Authorization ID     : GSTAGER
Connection Timestamp        : 2008-10-31T13:28:48.538973
Client Process ID           : 28167
Client Platform             : 30
Client Product ID           : SQL09070
Client Hostname             : gilera
Client Port Number          : 30143

UOW Level Details
-----
Start Time                  : 2008-10-31T13:28:51.560138
Stop Time                   : 2008-10-31T13:29:04.130849
Completion Status           : COMMIT
UOW ID                      : 5
Workload Occurrence ID     : 1
Workload Name               : SYSDEFAULTUSERWORKLOAD
Workload ID                 : 1
Client userid               :
Client Workstation Name     :
Client Application Name     :
Client Accounting String    :
Local Transaction ID        : 00000000000000EB
Global Transaction ID       : 0000000000000000000000000000000000000000
Log Space Used              : 0

UOW Metrics
-----
TOTAL_CPU_TIME              : 7459
TOTAL_WAIT_TIME             : 0
ACT_ABORTED_TOTAL          : 0
...

```

**Calculating the CPU time used by different applications or workloads with the unit of work event monitor:**

This topic shows one way that you can use the unit of work event monitor in day-to-day database operations.

In some business environments, departments are billed for the processing time their applications use. You can use the unit of work event to record the CPU time used by different application, workloads, or service classes. This information can, in turn, be used in accounting applications that perform billing for system resources.

**Before you begin**

The CREATE EVENT MONITOR statement requires a table space with a page size of at least 8 K to store the unformatted event (UE) table produced by the event monitor. Unless a table space is explicitly named in the CREATE EVENT MONITOR statement, the default table space for the database is used.

## About this task

This task describes a basic scenario for “charge-back” accounting. In the example that follows, all work performed on the system is tracked. From the data gathered, reports are created that show the CPU time used by different applications.

Depending on how your organization is set up, tracking system time based on workload might be appropriate. Alternatively, you can also look at the CPU time used in different service super classes, by specific workloads, or even by different users. If the data is written to relational tables, as the example in this task shows, you can use SQL to query and present the data in almost limitless ways.

**Note:** Activities within a unit of work can run in different service subclasses. For this reason, it is not appropriate to aggregate unit of work information by service subclass. If you want to aggregate CPU time by service class, use the activity event monitor instead.

## Procedure

1. Create a unit of work event monitor to capture information about units of work as they finish. For example, to create an event monitor called TRACKWORK, you might could use the following SQL:

```
CREATE EVENT MONITOR TRACKWORK FOR UNIT OF WORK WRITE TO UNFORMATTED EVENT TABLE
```

This statement creates a unit of work event monitor that writes to an unformatted event (UE) table. The UE table has the same name as the event monitor itself, TRACKWORK, and it is stored in the default table space.

2. Tell the database manager that you want to collect event information for all units of work completed on the database by running the following command:

```
UPDATE DATABASE CONFIGURATION FOR dbname USING MON_UOW_DATA BASE
```

This command causes information about all units of work executed on the data server to be sent to the active unit of work event monitors when the units of work complete. See “Configuring data collection” on page 164 for more information about controlling the scope of the unit of work data that is collected.

3. Next, activate the event monitor:

```
SET EVENT MONITOR TRACKWORK STATE 1
```

**Note:** By default, this event monitor starts automatically upon database activation, because the AUTOSTART option is applied by default. However, because this event monitor is being created in an already-active database, you must use the **SET EVENT MONITOR** command to start it manually.

From this point on, the unit of work event monitor captures information for each unit of work as it runs to completion. As each unit of work completes, the event monitor adds a record for the event to the UE table TRACKWORK.

4. When you are ready to collect data for reporting purposes, you must extract the records from the TRACKWORK UE table.

You can view this information in XML or relational format, using either the `EVMON_FORMAT_UE_TO_XML` or the `EVMON_FORMAT_UE_TO_TABLES` procedure to convert the data in the UE table. Alternatively, you can use the **db2evmonfmt** tool to create a text report of the information returned by the event monitor. This example shows the use of `EVMON_FORMAT_UE_TO_TABLES` to create relational tables that you can query in whatever way suits your needs.

```
CALL EVMON_FORMAT_UE_TO_TABLES
('UOW', NULL, NULL, NULL, NULL, NULL, NULL, -1, 'SELECT * FROM TRACKWORK')
```

The EVMON\_FORMAT\_UE\_TO\_TABLES procedure examines the UE table TRACKWORK produced by the event monitor; it selects each of the records from the UE table, and from them, creates rows containing the data collected by the unit of work event monitor in two relational tables:

- UOW\_EVENT
- UOW\_METRICS

The first table contains the most frequently used monitor elements and metrics associated with each event captured. The second contains detailed metrics for each event.

**Notes:**

- If you specify PKGLIST rather than BASE for the MON\_UOW\_DATA configuration parameter in step 2 on page 212, the EVMON\_FORMAT\_UE\_TO\_TABLES procedure creates a third table called UOW\_PACKAGE\_LIST. This table contains package list information related to the units of work. However, in this example, because only basic monitor elements are collected (see step 2 on page 212), this table will not contain any data. (See “Unit of work event monitor package listing information” on page 201 for more information about how the package list information can be used.)
  - The values in the columns of UOW\_METRICS can also be found in the XML document contained in the METRICS column of the UOW\_EVENT table. They are provided in the UOW\_METRICS table for more convenient, column-oriented access.
5. Query the tables produced in the previous step to see how CPU time was used by applications. The statement that follows returns a breakdown of total CPU time used by different users on the system since the unit of work event monitor was initialized. (This example assumes that client applications have identified themselves to the database using the sqlseti API, or through whatever application development environment you might be using, such as IBM Rational® Application Developer for WebSphere® Software.

```
SELECT SUBSTR(E.CLIENT_USERID,1,10) AS CLIENT_ID,
       SUBSTR(E.CLIENT_APPLNAME,1,80) AS CLIENT_APP,
       SUBSTR(E.CLIENT_WRKSTNNAME,1,10) AS WKSTN,
       SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
FROM UOW_EVENT E, UOW_METRICS M
WHERE M.APPLICATION_ID = E.APPLICATION_ID
      AND M.UOW_ID = E.UOW_ID
      AND M.MEMBER = E.MEMBER
GROUP BY E.CLIENT_USERID, E.CLIENT_APPLNAME, E.CLIENT_WRKSTNNAME
ORDER BY CPU_TIME DESC;
```

The preceding query returns the following results:

| CLIENT_ID | CLIENT_APP   | WKSTN | CPU_TIME  |
|-----------|--|-------|-----------|
|           |  |       | 987770013 |
|           | DB2BATCH   |       | 249375000 |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003021324173 |       | 91181678  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1004201047173 |       | 66097348  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191536588 |       | 28824420  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191536434 |       | 27555568  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221122075 |       | 16203116  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221118191 |       | 15759227  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221531062 |       | 15630121  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221117466 |       | 15236718  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221116141 |       | 14607249  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003251550366 |       | 14427883  |
| CLP       | C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003051054311 |       | 1312500   |

```

CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003051053301      1296875
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003051139066      1296875
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003051152281      1281250
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003041230283      1046875
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003291503479      1031250
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003291503479      515625
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003251506219      484375
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221444488      453125
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003021323249      406250
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003251544498      296875
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003171431559      171875
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003041227488      156250
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003221117188      109375
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003021333329      62500
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191502148      62500
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191527385      62500
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191528492      62500
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191530518      62500
CLP C:\DOCUME~1\ALLUSE~1\APPLIC~1\IBM\DB2\DB2COPY1\DB2\TMP\CCSCRIPT1003191533265      62500
CLP C:\Documents and Settings\A11 Users\Application Data\IBM\DB2\DB2COPY1\VB2DAS      62500

```

6. At this point, the unit of work event monitor TRACKWORK is still collecting information. Depending on how you want to track CPU time used by different applications, users or workloads, you can choose to take one of the following courses of action:

- If you want to calculate CPU usage on a daily basis, you can leave this unit of work event monitor active. Each day, run the EVMON\_FORMAT\_UE\_TO\_TABLES procedure to retrieve just the time-spent metrics for the preceding day:

```

CALL EVMON_FORMAT_UE_TO_TABLES
('UOW', NULL, NULL, NULL, NULL, NULL, NULL, -1,
 'SELECT * FROM TRACKWORK
  WHERE (DATE(EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))'
)

```

With this approach, the three relational tables produced by the EVMON\_FORMAT\_UE\_TO\_TABLES procedure continue to grow, providing a history of CPU usage over time. The query in step 5 on page 213 returns the cumulative totals for CPU time since the tables were first created with the EVMON\_FORMAT\_UE\_TO\_TABLES procedure. You can modify that query to show only the results from the previous day as follows:

```

SELECT SUBSTR(E.CLIENT_USERID,1,10) AS CLIENT_ID,
       SUBSTR(E.CLIENT_APPLNAME,1,80) AS CLIENT_APP,
       SUBSTR(E.CLIENT_WRKSTNNAME,1,10) AS WKSTN,
       SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
FROM UOW_EVENT E, UOW_METRICS M
WHERE M.APPLICATION_ID = E.APPLICATION_ID
      AND M.UOW_ID = E.UOW_ID
      AND M.MEMBER = E.MEMBER
      AND (DATE(E.EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))
GROUP BY E.CLIENT_USERID, E.CLIENT_APPLNAME, E.CLIENT_WRKSTNNAME
ORDER BY CPU_TIME DESC;

```

**Tip:** If you want to track CPU usage on a daily basis, but also want to manage how much data you collect on your system, remove data you no longer need from the UE table after you have updated the relational tables. For example, to delete the data collected on the previous day from the UE table TRACKWORK, use a DELETE statement similar to the one that follows:

```
DELETE FROM TRACKWORK WHERE (DATE(EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))
```

While an event monitor is active, it holds an intention exclusive (IX) table lock on any tables to which it writes information to prevent those tables from being dropped while it is using them. When a large number of rows is being deleted, the DELETE statement acquires a large number of row locks. In this

situation, lock escalation might occur, as row locks might be converted to a table lock. This request for table lock can cause the DELETE statement to hang, since the event monitor already has a lock on the table.

To avoid this situation, consider setting a lock timeout before issuing the DELETE statement:

```
SET CURRENT LOCK TIMEOUT 60
```

If increasing the lock timeout period does not resolve the problem, try deleting smaller subsets of the data, such as the records for smaller time periods (for example, 6 or 12 hours). This approach requires fewer locks, which will reduce the chance of lock escalation happening.

You can also prune the relational tables produced by `EVMON_FORMAT_UE_TO_TABLES` as needed to balance storage requirements with the need to view historical data.

- If you are finished calculating CPU time, you can stop the collection of event monitor information, and drop the event monitor and its related tables by performing the following steps:
  - a. Disable the collection of unit of work for this event monitor information using the **SET EVENT MONITOR TRACKWORK STATE 0** command.
  - b. Drop the event monitor itself using the `DROP EVENT MONITOR` statement.
  - c. Drop the tables related to the event monitor using a `DROP TABLE` statement. In this case, there are four tables in total to drop:
    - `TRACKWORK`, the UE table used to collect information from the event monitor
    - `UOW_EVENT`
    - `UOW_METRICS`
    - `UOW_PACKAGE_LIST`
  - d. Optional: If there are no remaining active event monitors, you might want to update the database configuration such that no unit of work event information is collected using the following command:

```
UPDATE DATABASE CONFIGURATION FOR dbname USING MON_UOW_DATA NONE
```

### Variation: Collecting metrics for specific workloads

The previous example illustrates how you can capture unit of work metrics for all work done on the system. Setting the scope of data collected using the **UPDATE DATABASE CONFIGURATION** command might cause more information to be collected than you need. You might, for example, want to track only work done by specific workloads. In this case, rather than enable collection of unit of work information across the whole database as shown in step 2 on page 212, you can specify the `COLLECT UNIT OF WORK DATA` clause with the `CREATE` or `ALTER WORKLOAD` statements. This clause causes only data for the workload specified to be collected by the event monitor. For example, to collect unit of work data for the workload named `PAYROLL`, use the following statement:

```
ALTER WORKLOAD PAYROLL COLLECT UNIT OF WORK DATA BASE
```

You can collect data for multiple workloads by running an `ALTER WORKLOAD` statement for each.

The remaining steps are the same, except for step 5 on page 213, where you would change the query to resemble the one that follows:

```
SELECT E.WORKLOAD_NAME,
       SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
       FROM UOW_EVENT E, UOW_METRICS M
WHERE  M.APPLICATION_ID = E.APPLICATION_ID
       AND M.UOW_ID = E.UOW_ID
       AND M.MEMBER = E.MEMBER
GROUP BY E.WORKLOAD_NAME
ORDER BY CPU_TIME DESC
```

The preceding statement reports the CPU time for each workload for which metrics collection is enabled:

| WORKLOAD  | CPU_TIME   |
|-----------|------------|
| PAYROLL   | 2143292042 |
| MARKETING | 492784916  |

2 record(s) selected.

## Package cache statement eviction event monitoring

The package cache event monitor captures data related to statement entries that have been flushed from the database package cache. This event monitor provides the history of the contents of the package cache which can help with SQL query performance and problem determination issues.

### Overview

The package cache event monitor collects the same information as the `MON_GET_PKG_CACHE_STMT` table function, including the full set of available activity metrics and the executable section information of an entry.

Starting in Version 10.1, you can get information about input arguments related to the longest-running statement. This statement is the one associated with the monitor element `max_coord_stmt_exec_time`. The input arguments associated with this statement are recorded as part of the `pkgcache_stmt_args` logical data group.

Two control mechanisms on the `CREATE EVENT MONITOR` statement help limit the volume of data that can be captured. The two control mechanisms provide the following capabilities:

- Filter entries with the `WHERE` clause based on one or more of the following conditions:
  - Whether the last update of the metrics for an entry occurs after a specific time before it is evicted (`UPDATED_SINCE_BOUNDARY_TIME`). An entry will only be collected if the time that the metrics were last updated is more recent than boundary time defined for the event monitor. The boundary time for an event monitor can be set using the `MON_GET_PKG_CACHE_STMT` table function. If no boundary time has been set for the event monitor, the `UPDATED_SINCE_BOUNDARY_TIME` clause will have no effect.
  - The number of times the section of an entry was executed (`NUM_EXECUTIONS`)
  - The total aggregated amount of time spent executing the statement (`STMT_EXEC_TIME`)
- `COLLECT DATA` clause options:
  - `COLLECT BASE DATA`

Same information collected as the MON\_GET\_PKG\_CACHE\_STMT table function, as well as the full set of available activity metrics

- **COLLECT DETAILED DATA**

Collects the same information gathered with the COLLECT BASE DATA clause and includes the executable section of the entry

When you need to investigate the individual execution of an SQL statement, you can use the MON\_GET\_PKG\_CACHE\_STMT table function (if the entries are still in the package cache) to compare the behavior of a cached entry relative to others. The execution metrics, compilation environment, and detailed descriptions for a cached entry are available for diagnostic purposes.

If an entry has already been flushed from the package cache, you can use the package cache event monitor to review the history of the cached entries which were flushed from the package cache. The history data contains the same information that the MON\_GET\_PKG\_CACHE\_STMT table function provides. In addition, the event monitor also provides the executable section of the statement. All of this applies to both dynamic and static SQL statements.

## Creating a package cache event monitor

To create a package cache event monitor and collect package cache event monitor data, you must have DBADM or SQLADM authority.

A package cache event monitor can write its output to either a regular table or an unformatted event table.

Before you create a package cache event monitor, identify the table space where you plan to store the output for your event monitor. The CREATE EVENT MONITOR statement will assume a default table space if you do not specify one. However, the recommended practice is to have a table space dedicated and configured to store the output table or tables associated with any event monitor. If you are using an unformatted event table, create package cache event monitors in table spaces with at least an 8K pagesize to ensure that event data is contained within the inlined BLOB column of the UE table. If the BLOB column is not inlined, then the performance of writing and reading the events to the unformatted event table might not be efficient.

To setup a package cache event monitor using defaults and best practices, complete the following steps:

- Create the event monitor by issuing the CREATE EVENT MONITOR statement. The following example uses defaults where possible and specifies to store the unformatted event table in an existing table space MY\_EVMON\_TABLESPACE:

```
CREATE EVENT MONITOR MY_PKG_CACHE_EVMON
FOR PACKAGE CACHE
WRITE TO UNFORMATTED EVENT TABLE (IN MY_EVMON_TABLESPACE)
```

## Enabling data collection

To enable data collection, you must activate the event monitor using the SET EVENT MONITOR STATE statement. The package cache event monitor is not a *passive* event monitor; following activation, it automatically starts collecting data whenever a statement is flushed from the package cache and meets the filter criteria set at the time of creation of the package cache event monitor.

## Accessing event data captured by a package cache event monitor

A unit of work event monitor can write data to a regular table or it can write data in binary format to an unformatted event (UE) table. You can access the data in regular tables by using SQL.

To access data in a UE table, use one of the following table functions:

### **EVMON\_FORMAT\_UE\_TO\_XML**

Extracts data from an unformatted event table into an XML document.

### **EVMON\_FORMAT\_UE\_TO\_TABLES**

Extracts data from an unformatted event table into a set of relational tables.

When you use one of these table functions, you can specify which data to extract by including a `SELECT` statement as one of the parameters to the function. You have full control over selection, ordering, and other aspects provided by the `SELECT` statement.

The schema file `~/sql11ib/misc/DB2EvmonPkgCache.xsd` is used to document the expected output of the package cache event monitor report in an XML document. The schema file will reference a common monitor schema file (`DB2MonCommon.xsd`) to avoid duplicating the common contents.

An XML stylesheet is provided in `~/sql11ib/samples/jdbc/DB2EvmonPkgCache.xsl`.

Use these table functions to specify the data to extract using a `SELECT` statement. You have full control over selection, ordering, and other aspects provided by the `SELECT` statement.

You can also use the `db2evmonfmt` command to perform the following tasks:

- Select events of interest based on the following attributes: executable ID, section type, query cost estimate, statement package cache ID, and flush time.
- Choose whether to receive the output in the form of a text report or a formatted XML document.
- Control the output format by creating your own XSLT style sheets instead of using the ones provided by the `db2evmonfmt` command.

For example, the following command provides a package cache report that:

1. Selects package cache events that have occurred in the past 24 hours in the database `SAMPLE`. These event records are obtained from the unformatted event table called `SAMPLE_PKGCACHE_EVENTS`.
2. Provides formatted text output using the `DB2EvmonPkgCache.xsl` style sheet.

```
java db2evmonfmt -d SAMPLE -ue SAMPLE_PKGCACHE_EVENTS -ftext -ss DB2EvmonPkgCache.xsl -hours 24
```

## Data generated by package cache event monitors

Package cache event monitors produce data about packages evicted from the package cache. You can choose to have the output from a package cache event monitor to regular tables, or to an unformatted event (UE) table. If data is written to a UE table, you must perform post-processing on it to view the data.

Regardless of the output format you choose, all package cache event data comes from one of three logical groups:

- pkgcache
- pkgcache\_metrics
- pkgcache\_stmt\_args

If you choose to have the package cache event data written to regular tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

**Note:** Unlike the locking and unit of work event monitors, you do not need to enable the generation of package cache event data after you create a package cache event monitor; data collection begins as soon as the event monitor is activated.

### Information written to tables for a package cache event monitor:

Information written by a package cache event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for a package cache event monitor, by default, three tables are produced, each containing monitor elements from one or more logical data groups.

*Table 39. Tables produced by package cache write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name                    | Logical data groups reported  |
|---------------------------------------|---|
| PKGCACHE_ <i>evmon-name</i>           | pkgcache  |
| PKGCACHE_METRICS_ <i>evmon-name</i>   | pkgcache_metrics  |
| PKGCACHE_STMT_ARGS_ <i>evmon-name</i> | “pkgcache_stmt_args logical data group” on page 79  |
| CONTROL_ <i>evmon-name</i>            | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statement. Refer to the reference topics for those statements for details.

### Tables produced

*Table 40. Information returned for a package cache event monitor: Default table name: PKGCACHE\_*evmon-name**

| Column name         | Data type     | Description   |
|---------------------|---------------|---|
| PARTITION_KEY       | INTEGER       | “partition_key - Partitioning key monitor element” on page 1016 |
| COMP_ENV_DESC       | BLOB          | comp_env_desc - Compilation environment                         |
| EFFECTIVE_ISOLATION | CHARACTER (2) | effective_isolation - Effective isolation                       |
| EVENT_ID            | BIGINT        | event_id - Event ID monitor element                             |
| EVENT_TIMESTAMP     | TIMESTAMP     | event_timestamp - Event timestamp monitor element               |

Table 40. Information returned for a package cache event monitor: Default table name: PKGCACHE\_evmon-name (continued)

| Column name                   | Data type        | Description  |
|-------------------------------|------------------|--|
| EXECUTABLE_ID                 | VARCHAR<br>(32)  | executable_id - Executable ID  |
| INSERT_TIMESTAMP              | TIMESTAMP        | insert_timestamp - Insert timestamp  |
| LAST_METRICS_UPDATE           | TIMESTAMP        | last_metrics_update - Metrics last update timestamp                                  |
| MAX_COORD_STMT_EXEC_TIME      | BIGINT           | max_coord_stmt_exec_time - Maximum coordinator statement execution time              |
| MAX_COORD_STMT_EXEC_TIMESTAMP | TIMESTAMP        | max_coord_stmt_exec_timestamp - Maximum coordinated statement execution timestamp    |
| MEMBER                        | SMALLINT         | member - Database member   |
| METRICS                       | BLOB             |  |
| NUM_COORD_EXEC                | BIGINT           | num_coord_exec - Number of executions by coordinator agent                           |
| NUM_COORD_EXEC_WITH_METRICS   | BIGINT           | num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics |
| NUM_EXEC_WITH_METRICS         | BIGINT           | num_exec_with_metrics - Number of executions with metrics collected                  |
| NUM_EXECUTIONS                | BIGINT           | num_executions - Statement executions  |
| PACKAGE_NAME                  | VARCHAR<br>(128) | package_name - Package name  |
| PACKAGE_SCHEMA                | VARCHAR<br>(128) | package_schema - Package schema  |
| PACKAGE_VERSION_ID            | VARCHAR<br>(64)  | package_version_id - Package version   |
| PREP_TIME                     | BIGINT           | prep_time - Preparation time   |
| QUERY_COST_ESTIMATE           | BIGINT           | query_cost_estimate - Query cost estimate  |
| QUERY_DATA_TAG_LIST           | VARCHAR<br>(32)  | query_data_tag_list - Query data tag list  |
| ROUTINE_ID                    | BIGINT           | routine_id - Routine ID  |
| SECTION_ENV                   | BLOB(0)          | section_env - Section environment  |
| SECTION_NUMBER                | BIGINT           | section_number - Section number  |
| SECTION_TYPE                  | CHARACTER<br>(1) | section_type - Section type indicator  |
| STMT_PKG_CACHE_ID             | BIGINT           | "stmt_pkgcache_id - Statement package cache identifier monitor element" on page 1251 |
| STMT_TEXT                     | CLOB             | stmt_text - SQL statement text   |
| STMT_TYPE_ID                  | VARCHAR<br>(32)  | stmt_type_id - Statement type identifier   |
| TOTAL_STATS_FABRICATION_TIME  | BIGINT           | total_stats_fabrication_time - Total statistics fabrication time                     |
| TOTAL_STATS_FABRICATIONS      | BIGINT           | total_stats_fabrications - Total statistics fabrications                             |
| TOTAL_SYNC_RUNSTATS           | BIGINT           | total_sync_runstats - Total synchronous RUNSTATS activities                          |
| TOTAL_SYNC_RUNSTATS_TIME      | BIGINT           | total_sync_runstats_time - Total synchronous RUNSTATS time                           |

Table 41. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS\_evmon-name

| Column name                  | Data type | Description  |
|------------------------------|-----------|--|
| PARTITION_KEY                | INTEGER   | "partition_key - Partitioning key monitor element" on page 1016        |
| EVENT_ID                     | BIGINT    | event_id - Event ID monitor element                                    |
| EVENT_TIMESTAMP              | TIMESTAMP | event_timestamp - Event timestamp monitor element                      |
| MEMBER                       | SMALLINT  | member - Database member   |
| WLM_QUEUE_TIME_TOTAL         | BIGINT    | wlm_queue_time_total - Workload manager total queue time               |
| WLM_QUEUE_ASSIGNMENTS_TOTAL  | BIGINT    | wlm_queue_assignments_total - Workload manager total queue assignments |
| FCM_TQ_RECV_WAIT_TIME        | BIGINT    | fcm_tq_recv_wait_time - FCM table queue received wait time             |
| FCM_MESSAGE_RECV_WAIT_TIME   | BIGINT    | fcm_message_recv_wait_time - FCM message received wait time            |
| FCM_TQ_SEND_WAIT_TIME        | BIGINT    | fcm_tq_send_wait_time - FCM table queue send wait time                 |
| FCM_MESSAGE_SEND_WAIT_TIME   | BIGINT    | fcm_message_send_wait_time - FCM message send wait time                |
| LOCK_WAIT_TIME               | BIGINT    | lock_wait_time - Time waited on locks                                  |
| LOCK_WAITS                   | BIGINT    | lock_waits - Lock waits  |
| DIRECT_READ_TIME             | BIGINT    | direct_read_time - Direct read time                                    |
| DIRECT_READ_REQS             | BIGINT    | direct_read_reqs - Direct read requests                                |
| DIRECT_WRITE_TIME            | BIGINT    | direct_write_time - Direct write time                                  |
| DIRECT_WRITE_REQS            | BIGINT    | direct_write_reqs - Direct write requests                              |
| LOG_BUFFER_WAIT_TIME         | BIGINT    | log_buffer_wait_time - Log buffer wait time                            |
| NUM_LOG_BUFFER_FULL          | BIGINT    | num_log_buffer_full - Number of full log buffers                       |
| LOG_DISK_WAIT_TIME           | BIGINT    | log_disk_wait_time - Log disk wait time                                |
| LOG_DISK_WAITS_TOTAL         | BIGINT    | log_disk_waits_total - Total log disk waits                            |
| POOL_WRITE_TIME              | BIGINT    | pool_write_time - Total buffer pool physical write time                |
| POOL_READ_TIME               | BIGINT    | pool_read_time - Total buffer pool physical read time                  |
| AUDIT_FILE_WRITE_WAIT_TIME   | BIGINT    | audit_file_write_wait_time - Audit file write wait time                |
| AUDIT_FILE_WRITES_TOTAL      | BIGINT    | audit_file_writes_total - Total audit files written                    |
| AUDIT_SUBSYSTEM_WAIT_TIME    | BIGINT    | audit_subsystem_wait_time - Audit subsystem wait time                  |
| AUDIT_SUBSYSTEM_WAITS_TOTAL  | BIGINT    | audit_subsystem_waits_total - Total audit subsystem waits              |
| DIAGLOG_WRITE_WAIT_TIME      | BIGINT    | diaglog_write_wait_time - Diagnostic log file write wait time          |
| DIAGLOG_WRITES_TOTAL         | BIGINT    | diaglog_writes_total - Total diagnostic log file writes                |
| FCM_SEND_WAIT_TIME           | BIGINT    | fcm_send_wait_time - FCM send wait time                                |
| FCM_RECV_WAIT_TIME           | BIGINT    | fcm_recv_wait_time - FCM received wait time                            |
| TOTAL_ACT_WAIT_TIME          | BIGINT    | total_act_wait_time - Total activity wait time                         |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT    | total_section_sort_proc_time - Total section sort processing time      |
| TOTAL_SECTION_SORTS          | BIGINT    | total_section_sorts - Total section sorts                              |
| TOTAL_SECTION_SORT_TIME      | BIGINT    | total_section_sort_time - Total section sort time                      |
| TOTAL_ACT_TIME               | BIGINT    | total_act_time - Total activity time                                   |
| ROWS_READ                    | BIGINT    | rows_read - Rows read  |
| ROWS_MODIFIED                | BIGINT    | rows_modified - Rows modified  |
| POOL_DATA_L_READS            | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                     |

Table 41. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS\_evmon-name (continued)

| Column name             | Data type | Description   |
|-------------------------|-----------|---|
| POOL_INDEX_L_READS      | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_TEMP_DATA_L_READS  | BIGINT    | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_INDEX_L_READS | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_XDA_L_READS        | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_TEMP_XDA_L_READS   | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| TOTAL_CPU_TIME          | BIGINT    | total_cpu_time - Total CPU time                                       |
| POOL_DATA_P_READS       | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_TEMP_DATA_P_READS  | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_XDA_P_READS        | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_TEMP_XDA_P_READS   | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS      | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_TEMP_INDEX_P_READS | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_DATA_WRITES        | BIGINT    | pool_data_writes - Buffer pool data writes                            |
| POOL_XDA_WRITES         | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| POOL_INDEX_WRITES       | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| DIRECT_READS            | BIGINT    | direct_reads - Direct reads from database                             |
| DIRECT_WRITES           | BIGINT    | direct_writes - Direct writes to database                             |
| ROWS_RETURNED           | BIGINT    | rows_returned - Rows returned   |
| DEADLOCKS               | BIGINT    | deadlocks - Deadlocks detected  |
| LOCK_TIMEOUTS           | BIGINT    | lock_timeouts - Number of lock timeouts                               |
| LOCK_ESCALS             | BIGINT    | lock_escals - Number of lock escalations                              |
| FCM_SENDS_TOTAL         | BIGINT    | fcm_sends_total - FCM sends total                                     |
| FCM_RECVS_TOTAL         | BIGINT    | fcm_recvs_total - FCM receives total                                  |
| FCM_SEND_VOLUME         | BIGINT    | fcm_send_volume - FCM send volume                                     |
| FCM_RECV_VOLUME         | BIGINT    | fcm_recv_volume - FCM received volume                                 |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT    | fcm_message_sends_total - Total FCM message sends                     |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT    | fcm_message_recvs_total - Total FCM message receives                  |
| FCM_MESSAGE_SEND_VOLUME | BIGINT    | fcm_message_send_volume - FCM message send volume                     |
| FCM_MESSAGE_RECV_VOLUME | BIGINT    | fcm_message_recv_volume - FCM message received volume                 |
| FCM_TQ_SENDS_TOTAL      | BIGINT    | fcm_tq_sends_total - FCM table queue send total                       |
| FCM_TQ_RECVS_TOTAL      | BIGINT    | fcm_tq_recvs_total - FCM table queue receives total                   |
| FCM_TQ_SEND_VOLUME      | BIGINT    | fcm_tq_send_volume - FCM table queue send volume                      |
| FCM_TQ_RECV_VOLUME      | BIGINT    | fcm_tq_recv_volume - FCM table queue received volume                  |
| TQ_TOT_SEND_SPILLS      | BIGINT    | tq_tot_send_spills - Total number of table queue buffers overflowed   |
| POST_THRESHOLD_SORTS    | BIGINT    | post_threshold_sorts - Post threshold sorts                           |
| POST_SHRTHRESHOLD_SORTS | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts                 |
| SORT_OVERFLOWS          | BIGINT    | sort_overflows - Sort overflows                                       |
| AUDIT_EVENTS_TOTAL      | BIGINT    | audit_events_total - Total audit events                               |

Table 41. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS\_evmon-name (continued)

| Column name                       | Data type | Description   |
|-----------------------------------|-----------|---|
| TOTAL_SORTS                       | BIGINT    | total_sorts - Total sorts   |
| STMT_EXEC_TIME                    | BIGINT    | stmt_exec_time - Statement execution time                                   |
| COORD_STMT_EXEC_TIME              | BIGINT    | coord_stmt_exec_time - Execution time for statement by coordinator agent    |
| TOTAL_ROUTINE_NON_SECT_PROC_TIME  | BIGINT    | total_routine_non_sect_proc_time - Non-section processing time              |
| TOTAL_ROUTINE_NON_SECT_TIME       | BIGINT    | total_routine_non_sect_time - Non-section routine execution time            |
| TOTAL_SECTION_PROC_TIME           | BIGINT    | total_section_proc_time - Total section processing time                     |
| TOTAL_APP_SECTION_EXECUTIONS      | BIGINT    | total_app_section_executions - Total application section executions         |
| TOTAL_SECTION_TIME                | BIGINT    | total_section_time - Total section time                                     |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT    | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME      | BIGINT    | total_routine_user_code_time - Total routine user code time                 |
| TOTAL_ROUTINE_TIME                | BIGINT    | total_routine_time - Total routine time                                     |
| THRESH_VIOLATIONS                 | BIGINT    | thresh_violations - Number of threshold violations                          |
| NUM_LW_THRESH_EXCEEDED            | BIGINT    | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded            |
| TOTAL_ROUTINE_INVOCATIONS         | BIGINT    | total_routine_invocations - Total routine invocations                       |
| LOCK_WAIT_TIME_GLOBAL             | BIGINT    | lock_wait_time_global - Lock wait time global                               |
| LOCK_WAITS_GLOBAL                 | BIGINT    | lock_waits_global - Lock waits global                                       |
| RECLAIM_WAIT_TIME                 | BIGINT    | reclaim_wait_time - Reclaim wait time                                       |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME    | BIGINT    | spacemappage_reclaim_wait_time - Space map page reclaim wait time           |
| LOCK_TIMEOUTS_GLOBAL              | BIGINT    | lock_timeouts_global - Lock timeouts global                                 |
| LOCK_ESCALS_MAXLOCKS              | BIGINT    | lock_escals_maxlocks - Number of maxlocks lock escalations                  |
| LOCK_ESCALS_LOCKLIST              | BIGINT    | lock_escals_locklist - Number of locklist lock escalations                  |
| LOCK_ESCALS_GLOBAL                | BIGINT    | lock_escals_global - Number of global lock escalations                      |
| CF_WAIT_TIME                      | BIGINT    | cf_wait_time - cluster caching facility wait time                           |
| CF_WAITS                          | BIGINT    | cf_waits - Number of cluster caching facility DB2 pureScale server waits    |
| POOL_DATA_GBP_L_READS             | BIGINT    | pool_data_gbp_l_reads - Group buffer pool data logical reads                |
| POOL_DATA_GBP_P_READS             | BIGINT    | pool_data_gbp_p_reads - Group buffer pool data physical reads               |
| POOL_DATA_LBP_PAGES_FOUND         | BIGINT    | pool_data_lbp_pages_found - Local buffer pool found data pages              |
| POOL_DATA_GBP_INVALID_PAGES       | BIGINT    | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages          |
| POOL_INDEX_GBP_L_READS            | BIGINT    | pool_index_gbp_l_reads - Group buffer pool index logical reads              |
| POOL_INDEX_GBP_P_READS            | BIGINT    | pool_index_gbp_p_reads - Group buffer pool index physical reads             |
| POOL_INDEX_LBP_PAGES_FOUND        | BIGINT    | pool_index_lbp_pages_found - Local buffer pool index pages found            |
| POOL_INDEX_GBP_INVALID_PAGES      | BIGINT    | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages        |

Table 41. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS\_evmon-name (continued)

| Column name                        | Data type | Description  |
|------------------------------------|-----------|--|
| POOL_XDA_GBP_L_READS               | BIGINT    | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests                        |
| POOL_XDA_GBP_P_READS               | BIGINT    | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests                       |
| POOL_XDA_LBP_PAGES_FOUND           | BIGINT    | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found                              |
| POOL_XDA_GBP_INVALID_PAGES         | BIGINT    | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages                          |
| EVMON_WAIT_TIME                    | BIGINT    | evmon_wait_time - Event monitor wait time  |
| EVMON_WAITS_TOTAL                  | BIGINT    | evmon_waits_total - Event monitor total waits  |
| TOTAL_EXTENDED_LATCH_WAIT_TIME     | BIGINT    | total_extended_latch_wait_time - Total extended latch wait time                                |
| TOTAL_EXTENDED_LATCH_WAITS         | BIGINT    | total_extended_latch_waits - Total extended latch waits  |
| TOTAL_DISP_RUN_QUEUE_TIME          | BIGINT    | total_disp_run_queue_time - Total dispatcher run queue time                                    |
| POOL_QUEUED_ASYNC_DATA_REQS        | BIGINT    | pool_queued_async_data_reqs - Data prefetch requests   |
| POOL_QUEUED_ASYNC_INDEX_REQS       | BIGINT    | pool_queued_async_index_reqs - Index prefetch requests   |
| POOL_QUEUED_ASYNC_XDA_REQS         | BIGINT    | pool_queued_async_xda_reqs - XDA prefetch requests   |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces           |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces         |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces        |
| POOL_QUEUED_ASYNC_OTHER_REQS       | BIGINT    | pool_queued_async_other_reqs - Non-prefetch requests   |
| POOL_QUEUED_ASYNC_DATA_PAGES       | BIGINT    | pool_queued_async_data_pages - Data pages prefetch requests                                    |
| POOL_QUEUED_ASYNC_INDEX_PAGES      | BIGINT    | pool_queued_async_index_pages - Index pages prefetch requests                                  |
| POOL_QUEUED_ASYNC_XDA_PAGES        | BIGINT    | pool_queued_async_xda_pages - XDA pages prefetch requests                                      |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES  | BIGINT    | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces    |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT    | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces  |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES   | BIGINT    | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS        | BIGINT    | pool_failed_async_data_reqs - Failed data prefetch requests                                    |
| POOL_FAILED_ASYNC_INDEX_REQS       | BIGINT    | pool_failed_async_index_reqs - Failed index prefetch requests                                  |
| POOL_FAILED_ASYNC_XDA_REQS         | BIGINT    | pool_failed_async_xda_reqs - Failed XDA prefetch requests                                      |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces    |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces  |

Table 41. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS\_evmon-name (continued)

| Column name                             | Data type | Description   |
|---|-----------|---|
| POOL_FAILED_ASYNC_TEMP_XDA_REQS         | BIGINT    | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces   |
| POOL_FAILED_ASYNC_OTHER_REQS            | BIGINT    | pool_failed_async_other_reqs - Failed non-prefetch requests   |
| TOTAL_PEDS                              | BIGINT    | total_peds - Total partial early distincts  |
| DISABLED_PEDS                           | BIGINT    | “disabled_peds - Disabled partial early distincts monitor element” on page 802  |
| POST_THRESHOLD_PEDS                     | BIGINT    | post_threshold_peds - Partial early distincts threshold   |
| TOTAL_PEAS                              | BIGINT    | total_peas - Total partial early aggregations   |
| POST_THRESHOLD_PEAS                     | BIGINT    | post_threshold_peas - Partial early aggregation threshold   |
| TQ_SORT_HEAP_REQUESTS                   | BIGINT    | tq_sort_heap_requests - Table queue sort heap requests  |
| TQ_SORT_HEAP_REJECTIONS                 | BIGINT    | tq_sort_heap_rejections - Table queue sort heap rejections  |
| PREFETCH_WAIT_TIME                      | BIGINT    | prefetch_wait_time - Time waited for prefetch   |
| PREFETCH_WAITS                          | BIGINT    | prefetch_waits - Prefetcher wait count  |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP  | BIGINT    | “pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element” on page 1041   |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | “pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element” on page 1073 |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP   | BIGINT    | “pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element” on page 1130     |

Table 42. Information returned for a package cache event monitor: Default table name: PKGCACHE\_STMT\_ARGS\_evmon-name

| Column name        | Data type     | Description   |
|--------------------|---------------|---|
| PARTITION_KEY      | INTEGER       | “partition_key - Partitioning key monitor element” on page 1016 |
| EVENT_ID           | BIGINT        | event_id - Event ID monitor element                             |
| EVENT_TIMESTAMP    | TIMESTAMP     | event_timestamp - Event timestamp monitor element               |
| MEMBER             | SMALLINT      | member - Database member  |
| STMT_VALUE_DATA    | CLOB          | stmt_value_data - Value data                                    |
| STMT_VALUE_INDEX   | INTEGER       | stmt_value_index - Value index                                  |
| STMT_VALUE_ISNULL  | INTEGER       | stmt_value_isnull - Value has null value                        |
| STMT_VALUE_ISREOPT | INTEGER       | stmt_value_isreopt - Variable used for statement reoptimization |
| STMT_VALUE_TYPE    | CHARACTER(16) | stmt_value_type - Value type                                    |

Entries for the following data types are recorded in the preceding table, however, the actual values of the arguments are not recorded in the STMT\_VALUE\_DATA element:

- BLOB
- CLOB
- REF

- BOOLEAN
- Structured data types
- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML tyoes
- DBCLOB
- ARRAY types
- ROW types
- ROWID
- CURSOR variables

Table 43. Information returned for a package cache event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | “partition_key - Partitioning key monitor element” on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

**Information written to relational tables by EVMON\_FORMAT\_UE\_TO\_TABLES for a package cache event monitor:**

Information written for a package cache event monitor from the EVMON\_FORMAT\_UE\_TO\_TABLES table function. This is also documented in the DB2EvmonPkgCache.xsd file.

Table 44. Information returned for a package cache event monitor: Table name: PKGCACHE\_EVENT

| Column Name        | Data Type                | Description   |
|--------------------|--------------------------|---|
| XMLID              | VARCHAR(256) NOT NULL    | “xmlid - XML ID monitor element” on page 1431           |
| EVENT_ID           | BIGINT NOT NULL          | event_id - Event ID monitor element                     |
| EVENT_TYPE         | VARCHAR(128) NOT NULL    | event_type - Event Type monitor element monitor element |
| EVENT_TIMESTAMP    | TIMESTAMP NOT NULL       | event_timestamp - Event timestamp monitor element       |
| MEMBER             | SMALLINT NOT NULL        | member - Database member                                |
| SECTION_TYPE       | CHAR(1)                  | section_type - Section type indicator                   |
| INSERT_TIMESTAMP   | TIMESTAMP                | insert_timestamp - Insert timestamp                     |
| EXECUTABLE_ID      | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID                           |
| PACKAGE_SCHEMA     | VARCHAR(128)             | package_schema - Package schema                         |
| PACKAGE_NAME       | VARCHAR(128)             | package_name - Package name                             |
| PACKAGE_VERSION_ID | VARCHAR(64)              | package_version_id - Package version                    |

Table 44. Information returned for a package cache event monitor: Table name: PKGCACHE\_EVENT (continued)

| Column Name                  | Data Type   | Description  |
|------------------------------|-------------|--|
| SECTION_NUMBER               | BIGINT      | section_number - Section number  |
| EFFECTIVE_ISOLATION          | CHAR(2)     | effective_isolation - Effective isolation  |
| NUM_EXECUTIONS               | BIGINT      | num_executions - Statement executions  |
| NUM_EXEC_WITH_METRICS        | BIGINT      | num_exec_with_metrics - Number of executions with metrics collected  |
| PREP_TIME                    | BIGINT      | prep_time - Preparation time   |
| LAST_METRICS_UPDATE          | TIMESTAMP   | last_metrics_update - Metrics last update timestamp  |
| NUM_COORD_EXEC               | BIGINT      | num_coord_exec - Number of executions by coordinator agent   |
| NUM_COORD_EXEC_WITH_METRICS  | BIGINT      | num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics   |
| STMT_TYPE_ID                 | VARCHAR(32) | stmt_type_id - Statement type identifier   |
| QUERY_COST_ESTIMATE          | BIGINT      | query_cost_estimate - Query cost estimate  |
| STMT_PKG_CACHE_ID            | BIGINT      | "stmt_pkgcache_id - Statement package cache identifier monitor element" on page 1251   |
| STMT_TEXT                    | CLOB(2M)    | stmt_text - SQL statement text   |
| COMP_ENV_DESC                | BLOB(10K)   | comp_env_desc - Compilation environment  |
| METRICS                      | BLOB(1M)    | XML document containing metrics-related monitor elements. The metrics in this document are the same as those described in the PKGCACHE_METRICS table that appears later in this topic. See "Interfaces that return monitor data in XML documents" on page 13 for more information. |
| SECTION_ENV                  | BLOB(150M)  | section_env - Section environment  |
| ROUTINE_ID                   | BIGINT      | routine_id - Routine ID  |
| QUERY_DATA_TAG_LIST          | VARCHAR(32) | query_data_tag_list - Query data tag list  |
| TOTAL_STATS_FABRICATION_TIME | BIGINT      | total_stats_fabrication_time - Total statistics fabrication time   |
| TOTAL_STATS_FABRICATIONS     | BIGINT      | total_stats_fabrications - Total statistics fabrications   |
| TOTAL_SYNC_RUNSTATS_TIME     | BIGINT      | total_sync_runstats_time - Total synchronous RUNSTATS time   |
| TOTAL_SYNC_RUNSTATS          | BIGINT      | total_sync_runstats - Total synchronous RUNSTATS activities  |

Table 44. Information returned for a package cache event monitor: Table name: PKGCACHE\_EVENT (continued)

| Column Name                   | Data Type | Description   |
|-------------------------------|-----------|---|
| MAX_COORD_STMT_EXEC_TIMESTAMP | TIMESTAMP | max_coord_stmt_exec_timestamp - Maximum coordinated statement execution timestamp |
| MAX_COORD_STMT_EXEC_TIME      | BIGINT    | max_coord_stmt_exec_time - Maximum coordinator statement execution time           |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table

| Column Name                  | Data Type             | Description   |
|------------------------------|-----------------------|---|
| XMLID                        | VARCHAR(256) NOT NULL | “xmlid - XML ID monitor element” on page 1431                     |
| TOTAL_ACT_TIME               | BIGINT                | total_act_time - Total activity time                              |
| TOTAL_ACT_WAIT_TIME          | BIGINT                | total_act_wait_time - Total activity wait time                    |
| TOTAL_CPU_TIME               | BIGINT                | total_cpu_time - Total CPU time                                   |
| POOL_READ_TIME               | BIGINT                | pool_read_time - Total buffer pool physical read time             |
| POOL_WRITE_TIME              | BIGINT                | pool_write_time - Total buffer pool physical write time           |
| DIRECT_READ_TIME             | BIGINT                | direct_read_time - Direct read time                               |
| DIRECT_WRITE_TIME            | BIGINT                | direct_write_time - Direct write time                             |
| LOCK_WAIT_TIME               | BIGINT                | lock_wait_time - Time waited on locks                             |
| TOTAL_SECTION_SORT_TIME      | BIGINT                | total_section_sort_time - Total section sort time                 |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT                | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS          | BIGINT                | total_section_sorts - Total section sorts                         |
| LOCK_ESCALS                  | BIGINT                | lock_escals - Number of lock escalations                          |
| LOCK_WAITS                   | BIGINT                | lock_waits - Lock waits   |
| ROWS_MODIFIED                | BIGINT                | rows_modified - Rows modified                                     |
| ROWS_READ                    | BIGINT                | rows_read - Rows read   |
| ROWS_RETURNED                | BIGINT                | rows_returned - Rows returned                                     |
| DIRECT_READS                 | BIGINT                | direct_reads - Direct reads from database                         |
| DIRECT_READ_REQS             | BIGINT                | direct_read_reqs - Direct read requests                           |
| DIRECT_WRITES                | BIGINT                | direct_writes - Direct writes to database                         |
| DIRECT_WRITE_REQS            | BIGINT                | direct_write_reqs - Direct write requests                         |
| POOL_DATA_L_READS            | BIGINT                | pool_data_l_reads - Buffer pool data logical reads                |
| POOL_TEMP_DATA_L_READS       | BIGINT                | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_XDA_L_READS             | BIGINT                | pool_xda_l_reads - Buffer pool XDA data logical reads             |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                 | Data Type | Description  |
|-----------------------------|-----------|--|
| POOL_TEMP_XDA_L_READS       | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads   |
| POOL_INDEX_L_READS          | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                   |
| POOL_TEMP_INDEX_L_READS     | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads    |
| POOL_DATA_P_READS           | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                    |
| POOL_TEMP_DATA_P_READS      | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads     |
| POOL_XDA_P_READS            | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                 |
| POOL_TEMP_XDA_P_READS       | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads  |
| POOL_INDEX_P_READS          | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                  |
| POOL_TEMP_INDEX_P_READS     | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads   |
| POOL_DATA_WRITES            | BIGINT    | pool_data_writes - Buffer pool data writes                             |
| POOL_XDA_WRITES             | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                          |
| POOL_INDEX_WRITES           | BIGINT    | pool_index_writes - Buffer pool index writes                           |
| TOTAL_SORTS                 | BIGINT    | total_sorts - Total sorts  |
| POST_THRESHOLD_SORTS        | BIGINT    | post_threshold_sorts - Post threshold sorts                            |
| POST_SHRTHRESHOLD_SORTS     | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts                  |
| SORT_OVERFLOWS              | BIGINT    | sort_overflows - Sort overflows  |
| WLM_QUEUE_TIME_TOTAL        | BIGINT    | wlm_queue_time_total - Workload manager total queue time               |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT    | wlm_queue_assignments_total - Workload manager total queue assignments |
| DEADLOCKS                   | BIGINT    | deadlocks - Deadlocks detected   |
| FCM_RECV_VOLUME             | BIGINT    | fcm_recv_volume - FCM received volume                                  |
| FCM_RECVS_TOTAL             | BIGINT    | fcm_recvs_total - FCM receives total                                   |
| FCM_SEND_VOLUME             | BIGINT    | fcm_send_volume - FCM send volume                                      |
| FCM_SENDS_TOTAL             | BIGINT    | fcm_sends_total - FCM sends total                                      |
| FCM_RECV_WAIT_TIME          | BIGINT    | fcm_recv_wait_time - FCM received wait time                            |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                      | Data Type | Description  |
|----------------------------------|-----------|--|
| FCM_SEND_WAIT_TIME               | BIGINT    | fcm_send_wait_time - FCM send wait time                                  |
| LOCK_TIMEOUTS                    | BIGINT    | lock_timeouts - Number of lock timeouts                                  |
| LOG_BUFFER_WAIT_TIME             | BIGINT    | log_buffer_wait_time - Log buffer wait time                              |
| NUM_LOG_BUFFER_FULL              | BIGINT    | num_log_buffer_full - Number of full log buffers                         |
| LOG_DISK_WAIT_TIME               | BIGINT    | log_disk_wait_time - Log disk wait time                                  |
| LOG_DISK_WAITS_TOTAL             | BIGINT    | log_disk_waits_total - Total log disk waits                              |
| TOTAL_ROUTINE_TIME               | BIGINT    | total_routine_time - Total routine time                                  |
| TOTAL_ROUTINE_INVOCATIONS        | BIGINT    | total_routine_invocations - Total routine invocations                    |
| COORD_STMT_EXEC_TIME             | BIGINT    | coord_stmt_exec_time - Execution time for statement by coordinator agent |
| STMT_EXEC_TIME                   | BIGINT    | stmt_exec_time - Statement execution time                                |
| TOTAL_SECTION_TIME               | BIGINT    | total_section_time - Total section time                                  |
| TOTAL_SECTION_PROC_TIME          | BIGINT    | total_section_proc_time - Total section processing time                  |
| TOTAL_ROUTINE_NON_SECT_TIME      | BIGINT    | total_routine_non_sect_time - Non-section routine execution time         |
| TOTAL_ROUTINE_NON_SECT_PROC_TIME | BIGINT    | total_routine_non_sect_proc_time - Non-section processing time           |
| FCM_TQ_RECV_WAIT_TIME            | BIGINT    | fcm_tq_recv_wait_time - FCM table queue received wait time               |
| FCM_MESSAGE_RECV_WAIT_TIME       | BIGINT    | fcm_message_recv_wait_time - FCM message received wait time              |
| FCM_TQ_SEND_WAIT_TIME            | BIGINT    | fcm_tq_send_wait_time - FCM table queue send wait time                   |
| FCM_MESSAGE_SEND_WAIT_TIME       | BIGINT    | fcm_message_send_wait_time - FCM message send wait time                  |
| AUDIT_FILE_WRITE_WAIT_TIME       | BIGINT    | audit_file_write_wait_time - Audit file write wait time                  |
| AUDIT_FILE_WRITES_TOTAL          | BIGINT    | audit_file_writes_total - Total audit files written                      |
| AUDIT_SUBSYSTEM_WAIT_TIME        | BIGINT    | audit_subsystem_wait_time - Audit subsystem wait time                    |
| AUDIT_SUBSYSTEM_WAITS_TOTAL      | BIGINT    | audit_subsystem_waits_total - Total audit subsystem waits                |
| DIAGLOG_WRITE_WAIT_TIME          | BIGINT    | diaglog_write_wait_time - Diagnostic log file write wait time            |
| DIAGLOG_WRITES_TOTAL             | BIGINT    | diaglog_writes_total - Total diagnostic log file writes                  |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                       | Data Type | Description   |
|-----------------------------------|-----------|---|
| FCM_MESSAGE_SENDS_TOTAL           | BIGINT    | fcm_message_sends_total - Total FCM message sends                           |
| FCM_MESSAGE_RECVS_TOTAL           | BIGINT    | fcm_message_recvs_total - Total FCM message receives                        |
| FCM_MESSAGE_SEND_VOLUME           | BIGINT    | fcm_message_send_volume - FCM message send volume                           |
| FCM_MESSAGE_RECV_VOLUME           | BIGINT    | fcm_message_recv_volume - FCM message received volume                       |
| FCM_TQ_SENDS_TOTAL                | BIGINT    | fcm_tq_sends_total - FCM table queue send total                             |
| FCM_TQ_RECVS_TOTAL                | BIGINT    | fcm_tq_recvs_total - FCM table queue receives total                         |
| FCM_TQ_SEND_VOLUME                | BIGINT    | fcm_tq_send_volume - FCM table queue send volume                            |
| FCM_TQ_RECV_VOLUME                | BIGINT    | fcm_tq_recv_volume - FCM table queue received volume                        |
| TQ_TOT_SEND_SPILLS                | BIGINT    | tq_tot_send_spills - Total number of table queue buffers overflowed         |
| AUDIT_EVENTS_TOTAL                | BIGINT    | audit_events_total - Total audit events                                     |
| TOTAL_APP_SECTION_EXECUTIONS      | BIGINT    | total_app_section_executions - Total application section executions         |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT    | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME      | BIGINT    | total_routine_user_code_time - Total routine user code time                 |
| THRESH_VIOLATIONS                 | BIGINT    | thresh_violations - Number of threshold violations                          |
| NUM_LW_THRESH_EXCEEDED            | BIGINT    | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded            |
| LOCK_WAITS_GLOBAL                 | BIGINT    | lock_waits_global - Lock waits global                                       |
| LOCK_WAIT_TIME_GLOBAL             | BIGINT    | lock_wait_time_global - Lock wait time global                               |
| LOCK_TIMEOUTS_GLOBAL              | BIGINT    | lock_timeouts_global - Lock timeouts global                                 |
| LOCK_ESCALS_MAXLOCKS              | BIGINT    | lock_escals_maxlocks - Number of maxlocks lock escalations                  |
| LOCK_ESCALS_LOCKLIST              | BIGINT    | lock_escals_locklist - Number of locklist lock escalations                  |
| LOCK_ESCALS_GLOBAL                | BIGINT    | lock_escals_global - Number of global lock escalations                      |
| RECLAIM_WAIT_TIME                 | BIGINT    | reclaim_wait_time - Reclaim wait time                                       |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME    | BIGINT    | spacemappage_reclaim_wait_time - Space map page reclaim wait time           |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                    | Data Type | Description  |
|--------------------------------|-----------|--|
| CF_WAITS                       | BIGINT    | cf_waits - Number of cluster caching facility DB2 pureScale server waits       |
| CF_WAIT_TIME                   | BIGINT    | cf_wait_time - cluster caching facility wait time                              |
| POOL_DATA_GBP_L_READS          | BIGINT    | pool_data_gbp_l_reads - Group buffer pool data logical reads                   |
| POOL_DATA_GBP_P_READS          | BIGINT    | pool_data_gbp_p_reads - Group buffer pool data physical reads                  |
| POOL_DATA_LBP_PAGES_FOUND      | BIGINT    | pool_data_lbp_pages_found - Local buffer pool found data pages                 |
| POOL_DATA_GBP_INVALID_PAGES    | BIGINT    | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages             |
| POOL_INDEX_GBP_L_READS         | BIGINT    | pool_index_gbp_l_reads - Group buffer pool index logical reads                 |
| POOL_INDEX_GBP_P_READS         | BIGINT    | pool_index_gbp_p_reads - Group buffer pool index physical reads                |
| POOL_INDEX_LBP_PAGES_FOUND     | BIGINT    | pool_index_lbp_pages_found - Local buffer pool index pages found               |
| POOL_INDEX_GBP_INVALID_PAGES   | BIGINT    | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages           |
| POOL_XDA_GBP_L_READS           | BIGINT    | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests        |
| POOL_XDA_GBP_P_READS           | BIGINT    | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests       |
| POOL_XDA_LBP_PAGES_FOUND       | BIGINT    | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found              |
| POOL_XDA_GBP_INVALID_PAGES     | BIGINT    | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages          |
| EVMON_WAIT_TIME                | BIGINT    | evmon_wait_time - Event monitor wait time                                      |
| EVMON_WAITS_TOTAL              | BIGINT    | evmon_waits_total - Event monitor total waits                                  |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT    | total_extended_latch_wait_time - Total extended latch wait time                |
| TOTAL_EXTENDED_LATCH_WAITS     | BIGINT    | total_extended_latch_waits - Total extended latch waits                        |
| TOTAL_DISP_RUN_QUEUE_TIME      | BIGINT    | total_disp_run_queue_time - Total dispatcher run queue time                    |
| TOTAL_PEDS                     | BIGINT    | total_peds - Total partial early distincts                                     |
| DISABLED_PEDS                  | BIGINT    | “disabled_peds - Disabled partial early distincts monitor element” on page 802 |
| POST_THRESHOLD_PEDS            | BIGINT    | post_threshold_peds - Partial early distincts threshold                        |
| TOTAL_PEAS                     | BIGINT    | total_peas - Total partial early aggregations                                  |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                        | Data Type | Description  |
|------------------------------------|-----------|--|
| POST_THRESHOLD_PEAS                | BIGINT    | post_threshold_peas - Partial early aggregation threshold                                      |
| TQ_SORT_HEAP_REQUESTS              | BIGINT    | tq_sort_heap_requests - Table queue sort heap requests   |
| TQ_SORT_HEAP_REJECTIONS            | BIGINT    | tq_sort_heap_rejections - Table queue sort heap rejections                                     |
| POOL_QUEUED_ASYNC_DATA_REQS        | BIGINT    | pool_queued_async_data_reqs - Data prefetch requests   |
| POOL_QUEUED_ASYNC_INDEX_REQS       | BIGINT    | pool_queued_async_index_reqs - Index prefetch requests   |
| POOL_QUEUED_ASYNC_XDA_REQS         | BIGINT    | pool_queued_async_xda_reqs - XDA prefetch requests   |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces           |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces         |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces        |
| POOL_QUEUED_ASYNC_OTHER_REQS       | BIGINT    | pool_queued_async_other_reqs - Non-prefetch requests   |
| POOL_QUEUED_ASYNC_DATA_PAGES       | BIGINT    | pool_queued_async_data_pages - Data pages prefetch requests                                    |
| POOL_QUEUED_ASYNC_INDEX_PAGES      | BIGINT    | pool_queued_async_index_pages - Index pages prefetch requests                                  |
| POOL_QUEUED_ASYNC_XDA_PAGES        | BIGINT    | pool_queued_async_xda_pages - XDA pages prefetch requests                                      |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES  | BIGINT    | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces    |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT    | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces  |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES   | BIGINT    | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS        | BIGINT    | pool_failed_async_data_reqs - Failed data prefetch requests                                    |
| POOL_FAILED_ASYNC_INDEX_REQS       | BIGINT    | pool_failed_async_index_reqs - Failed index prefetch requests                                  |
| POOL_FAILED_ASYNC_XDA_REQS         | BIGINT    | pool_failed_async_xda_reqs - Failed XDA prefetch requests                                      |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces    |

Table 45. Information returned for a package cache event monitor: Table name: PKGCACHE\_METRICS. The metrics in this table are the same as those returned in the METRICS monitor element in the PKGCACHE\_EVENT table (continued)

| Column Name                             | Data Type | Description   |
|---|-----------|---|
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS       | BIGINT    | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces   |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS         | BIGINT    | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces   |
| POOL_FAILED_ASYNC_OTHER_REQS            | BIGINT    | pool_failed_async_other_reqs - Failed non-prefetch requests   |
| PREFETCH_WAIT_TIME                      | BIGINT    | prefetch_wait_time - Time waited for prefetch   |
| PREFETCH_WAITS                          | BIGINT    | prefetch_waits - Prefetcher wait count  |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP  | BIGINT    | "pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element" on page 1041   |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | "pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element" on page 1073 |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP   | BIGINT    | "pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element" on page 1130     |

Table 46. Information returned for a package cache event monitor: Table name: PKGCACHE\_STMT\_ARGS.

| Column Name        | Data Type             | Description   |
|--------------------|-----------------------|---|
| XMLID              | VARCHAR(256) NOT NULL | "xmlid - XML ID monitor element" on page 1431                   |
| STMT_VALUE_INDEX   | INTEGER NOT NULL      | stmt_value_index - Value index                                  |
| STMT_VALUE_ISREOPT | INTEGER               | stmt_value_isreopt - Variable used for statement reoptimization |
| STMT_VALUE_ISNULL  | INTEGER               | stmt_value_isnull - Value has null value                        |
| STMT_VALUE_TYPE    | CHAR(16)              | stmt_value_type - Value type                                    |
| STMT_VALUE_DATA    | CLOB(32K)             | stmt_value_data - Value data                                    |

Entries for the following data types are recorded in the preceding table, however, the actual values of the arguments are not recorded in the STMT\_VALUE\_DATA element:

- BLOB
- CLOB
- REF
- BOOLEAN

- Structured data types
- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML tyoes
- DBCLOB
- ARRAY types
- ROW types
- ROWID
- CURSOR variables

Input arguments are recorded beginning with the one that appears first in the statement, and continuing with each one in succession. The number of input parameters that can be recorded in this table is constrained only by the upper limit on the size of the BLOB document that the UE event monitor uses to capture event information. In practical terms, it is unlikely the number input arguments captured would ever cause this limit to be reached.

#### Information written to XML for a package cache event monitor:

Information written for a package cache event monitor from the EVMON\_FORMAT\_UE\_TO\_XML table function. This is also documented in the DB2EvmonPkgCache.xsd file.

#### db2\_pkgcache\_event

The main schema that describes a package cache event event in details.

**Element content:** ( "section\_type" on page 236, "insert\_timestamp" on page 236, "executable\_id" on page 236, "package\_schema" on page 236, "package\_name" on page 236, "package\_version\_id" on page 237, "section\_number" on page 237 {zero or one times (?)}, "effective\_isolation" on page 237, "num\_executions" on page 237, "num\_exec\_with\_metrics" on page 237, "prep\_time" on page 238, "last\_metrics\_update" on page 238, "num\_coord\_exec" on page 238, "num\_coord\_exec\_with\_metrics" on page 238, "stmt\_type\_id" on page 239, "query\_cost\_estimate" on page 239, "stmt\_pkg\_cache\_id" on page 239, "stmt\_text" on page 239, "comp\_env\_desc" on page 240, "section\_env" on page 240, "activity\_metrics" on page 240, "routine\_id" on page 240, "query\_data\_tag\_list" on page 240, "total\_stats\_fabrication\_time" on page 240, "total\_stats\_fabrications" on page 241, "total\_sync\_runstats\_time" on page 241, "total\_sync\_runstats" on page 241, "max\_coord\_stmt\_exec\_timestamp" on page 241 {zero or one times (?)}, "max\_coord\_stmt\_exec\_time\_arg" on page 242 {zero or more (\*)}, "max\_coord\_stmt\_exec\_time" on page 242, ANY content ( skip ) {zero or more (\*)} )

#### Attributes:

| QName     | Type        | Fixed | Default | Use      | Annotation |
|-----------|-------------|-------|---------|----------|------------|
| id        | xs:long     |       |         | required |            |
| type      |             |       |         | required |            |
| timestamp | xs:dateTime |       |         | required |            |
| member    |             |       |         | required |            |
| release   | xs:long     |       |         | required |            |

| QName                            | Type | Fixed | Default | Use | Annotation |
|----------------------------------|------|-------|---------|-----|------------|
| ANY attribute from ANY namespace |      |       |         |     |            |

### section\_type

The type of SQL statement processed. Possible values: D:Dynamic or S:Static. See monitor element “section\_type - Section type indicator monitor element” on page 1202 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### insert\_timestamp

The time when the variation or section was inserted into the cache. See monitor element “insert\_timestamp - Insert timestamp monitor element” on page 878 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### executable\_id

A binary token generated on the data server that uniquely identifies the SQL statement section that was executed. See monitor element “executable\_id - Executable ID monitor element” on page 818 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### package\_schema

The schema name of the package associated with an SQL statement. See monitor element “package\_schema - Package schema monitor element” on page 1007 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### package\_name

The name of the package that contains the SQL statement currently executing. See monitor element “package\_name - Package name monitor element” on page 1006 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### package\_version\_id

The package version identifies the version identifier of the package that contains the SQL statement currently executing. See monitor element “package\_version\_id - Package version monitor element” on page 1007for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### section\_number

The internal section number in the package for the SQL statement currently processing or most recently processed. See monitor element “section\_number - Section number monitor element” on page 1201for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### effective\_isolation

The isolation value in effect for the SQL statement while it was being run. See monitor element “effective\_isolation - Effective isolation monitor element” on page 805for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |

### num\_executions

The number times the SQL statement has been executed. See monitor element “num\_executions - Statement executions monitor element” on page 973for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### num\_exec\_with\_metrics

The number times the SQL statement has been executed with the metrics. collected. See monitor element “num\_exec\_with\_metrics - Number of executions with metrics collected monitor element” on page 974for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### prep\_time

Time in milliseconds required to prepare an SQL statement if the activity is an SQL statement. See monitor element "prep\_time - Preparation time monitor element" on page 1154for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### last\_metrics\_update

Timestamp reflecting the last time metrics were updated for this cache entry. See monitor element "last\_metrics\_update - Metrics last update timestamp monitor element" on page 895for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### num\_coord\_exec

The number of times this section was executed by a coordinator agent. See monitor element "num\_coord\_exec - Number of executions by coordinator agent monitor element" on page 972for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### num\_coord\_exec\_with\_metrics

The number of times this section was executed by a coordinator agent and monitoring metrics were being captured. See monitor element "num\_coord\_exec\_with\_metrics - Number of executions by coordinator agent with metrics monitor element" on page 972for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### stmt\_type\_id

Statement type identifier. See monitor element "stmt\_type\_id - Statement type identifier monitor element" on page 1257for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Attributes:

| QName | Type    | Fixed | Default | Use      | Annotation |
|-------|---------|-------|---------|----------|------------|
| id    | xs:long |       |         | optional |            |

#### query\_cost\_estimate

Estimated cost for a query, as determined by the SQL compiler. See monitor element "query\_cost\_estimate - Query cost estimate monitor element" on page 1163for more details.

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### stmt\_pkg\_cache\_id

Contained by: "db2\_pkgcache\_event" on page 235

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### stmt\_text

The text of the SQL statement. See monitor element "stmt\_text - SQL statement text monitor element" on page 1255for more details.

Contained by: "db2\_pkgcache\_event" on page 235

### **comp\_env\_desc**

“comp\_env\_desc - Compilation environment monitor element” on page 722

**Contained by:** “db2\_pkgcache\_event” on page 235

### **section\_env**

A BLOB that contains the section for an SQL statement. See monitor element “section\_env - Section environment monitor element” on page 1201 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### **activity\_metrics**

The activity metrics for this cache entry.

**Contained by:** “db2\_pkgcache\_event” on page 235

### **routine\_id**

For CALL statements this element stores the routine identifier associated with the stored procedure invoked. See monitor element “routine\_id - Routine ID monitor element” on page 1187 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **query\_data\_tag\_list**

The data tag list for this cache entry. See monitor element “query\_data\_tag\_list - Estimated query data tag list monitor element” on page 1164 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

### **total\_stats\_fabrication\_time**

Total time spent, in milliseconds, on statistics fabrications by real-time statistics gathering. See monitor element “total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378 for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_stats\_fabrications**

Total number of statistics fabrications performed by real-time statistics during query compilation. See monitor element “total\_stats\_fabrications - Total statistics fabrications monitor elements” on page 1379for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_sync\_runstats\_time**

Total time spent, in milliseconds, on synchronous RUNSTATS activities triggered by real-time statistics gathering. See monitor element “total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_sync\_runstats**

Total number of synchronous RUNSTATS activities triggered by real-time statistics gathering. See monitor element “total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element” on page 1382for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **max\_coord\_stmt\_exec\_timestamp**

See monitor element “max\_coord\_stmt\_exec\_timestamp - Maximum coordinator statement execution timestamp monitor element” on page 945for more details.

**Contained by:** “db2\_pkgcache\_event” on page 235

#### **Element content:**

| Type        | Facet |
|-------------|-------|
| xs:dateTime |       |

### max\_coord\_stmt\_exec\_time\_arg

Describes an input variable to this statement which produced the max\_coord\_stmt\_exec\_time.

**Contained by:** "db2\_pkgcache\_event" on page 235

**Element content:** ( "stmt\_value\_index", "stmt\_value\_isreopt", "stmt\_value\_isnull", "stmt\_value\_type" on page 243, "stmt\_value\_data" on page 243, ANY content ( skip ) {zero or more (\*)} )

### max\_coord\_stmt\_exec\_time

See monitor element "max\_coord\_stmt\_exec\_time - Maximum coordinator statement execution time monitor element" on page 943for more details.

**Contained by:** "db2\_pkgcache\_event" on page 235

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### stmt\_value\_index

The element represents the position of the input parameter marker or host variable used in the SQL statement. See monitor element "stmt\_value\_index - Value index" on page 1259for more details.

**Contained by:** "max\_coord\_stmt\_exec\_time\_arg"

### stmt\_value\_isreopt

The element shows whether the variable was used during statement reoptimization. See monitor element "stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261for more details.

**Contained by:** "max\_coord\_stmt\_exec\_time\_arg"

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

### stmt\_value\_isnull

The element shows whether a data value associated with the SQL statement is the NULL value. See monitor element "stmt\_value\_isnull - Value has null value monitor element" on page 1260for more details.

**Contained by:** "max\_coord\_stmt\_exec\_time\_arg"

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

### stmt\_value\_type

“stmt\_value\_type - Value type monitor element” on page 1261

**Contained by:** “max\_coord\_stmt\_exec\_time\_arg” on page 242

### stmt\_value\_data

The element contains a string representation of a data value associated with an SQL statement. See monitor element “stmt\_value\_data - Value data” on page 1259 for more details.

**Contained by:** “max\_coord\_stmt\_exec\_time\_arg” on page 242

## Collecting package cache event data and generating reports

You can use the package cache event monitor to collect data about statement entries that were flushed from the database package cache. After the package cache event data has been collected in an unformatted event table, follow the directions in this task to obtain a text report.

### Before you begin

To collect package cache event monitor data, you must have DBADM or SQLADM authority.

### About this task

The package cache event monitor collects relevant history information about what was in the package cache to help with query performance and problem determination issues related to SQL statements. For example, some of the information the package cache event monitor collects from the database package cache is as follows:

- Executable ID (EXECUTABLE\_ID)
- The estimated cost of the query (QUERY\_COST\_ESTIMATE)
- The time that the entry was flushed from the package cache (Event Timestamp)

This task provides instructions for collecting package cache event data.

### Restrictions

Input data values are not viewable if you do not have DBADM or SQLADM authority.

### Procedure

To collect detailed information regarding package cache events, perform the following steps:

1. Create a package cache event monitor called cachesmtmtevm on by using the CREATE EVENT MONITOR FOR PACKAGE CACHE statement, as shown in the following example:



```

FCM_MESSAGE_SEND_WAIT_TIME      : 0
LOCK_WAIT_TIME                  : 0
LOCK_WAITS                      : 0
DIRECT_READ_TIME                : 0
DIRECT_READ_REQS                : 0
DIRECT_WRITE_TIME               : 3
DIRECT_WRITE_REQS               : 1
LOG_BUFFER_WAIT_TIME           : 0
NUM_LOG_BUFFER_FULL             : 0
LOG_DISK_WAIT_TIME              : 0
LOG_DISK_WAITS_TOTAL            : 0
POOL_WRITE_TIME                 : 0
POOL_READ_TIME                  : 33
AUDIT_FILE_WRITE_WAIT_TIME      : 0
AUDIT_FILE_WRITES_TOTAL         : 0
AUDIT_SUBSYSTEM_WAIT_TIME       : 0
AUDIT_SUBSYSTEM_WAITS_TOTAL     : 0
DIAGLOG_WRITE_WAIT_TIME         : 0
DIAGLOG_WRITES_TOTAL            : 0
FCM_SEND_WAIT_TIME              : 0
FCM_RECVD_WAIT_TIME             : 0
TOTAL_ACT_WAIT_TIME             : 36
TOTAL_SECTION_SORT_PROC_TIME    : 0
TOTAL_SECTION_SORTS              : 0
TOTAL_SECTION_SORT_TIME         : 0
TOTAL_ACT_TIME                   : 37
TOTAL_ROUTINE_TIME               : 0
STMT_EXEC_TIME                  : 3658
COORD_STMT_EXEC_TIME            : 3658
TOTAL_ROUTINE_NON_SECTION_PROC_TIME : 0
TOTAL_ROUTINE_NON_SECTION_TIME  : 0
TOTAL_SECTION_PROC_TIME         : 1
TOTAL_SECTION_TIME              : 37
TOTAL_ROUTINE_USER_CODE_PROC_TIME : 0
TOTAL_ROUTINE_USER_CODE_TIME    : 0
ROWS_READ                       : 19
ROWS_MODIFIED                   : 3
POOL_DATA_L_READS               : 42
POOL_INDEX_L_READS              : 83
POOL_TEMP_DATA_L_READS          : 0
POOL_TEMP_INDEX_L_READS        : 0
POOL_XDA_L_READS                : 0
POOL_TEMP_XDA_L_READS           : 0
TOTAL_CPU_TIME                   : 2243
POOL_DATA_P_READS               : 13
POOL_TEMP_DATA_P_READS          : 0
POOL_XDA_P_READS                : 0
POOL_TEMP_XDA_P_READS           : 0
POOL_INDEX_P_READS              : 33
POOL_TEMP_INDEX_P_READS         : 0
POOL_DATA_WRITES                 : 0
POOL_XDA_WRITES                  : 0
POOL_INDEX_WRITES                : 0
DIRECT_READS                     : 0
DIRECT_WRITES                     : 2
ROWS_RETURNED                    : 0
DEADLOCKS                        : 0
LOCK_TIMEOUTS                    : 0
LOCK_ESCALS                      : 0
FCM SENDS_TOTAL                  : 0
FCM_RECVS_TOTAL                  : 0
FCM_SEND_VOLUME                  : 0
FCM_RECV_VOLUME                  : 0
FCM_MESSAGE_SENDS_TOTAL          : 0
FCM_MESSAGE_RECVS_TOTAL          : 0
FCM_MESSAGE_SEND_VOLUME          : 0
FCM_MESSAGE_RECV_VOLUME          : 0
FCM_TQ_SENDS_TOTAL               : 0
FCM_TQ_RECVS_TOTAL               : 0
FCM_TQ_SEND_VOLUME               : 0
FCM_TQ_RECV_VOLUME               : 0
TQ_TOT_SEND_SPILLS               : 0
POST_THRESHOLD_SORTS             : 0
POST_SHRTHRESHOLD_SORTS         : 0
SORT_OVERFLOW                     : 0
AUDIT_EVENTS_TOTAL               : 0
TOTAL_SORTS                       : 0
THRESH_VIOLATIONS                : 0
NUM_LW_THRESH_EXCEEDED           : 0
TOTAL_ROUTINE_INVOCATIONS        : 0

```

**Using package cache information to identify statements that are candidates for performance tuning:**

You can use the package cache event monitor along with in-memory metrics to identify which statements from the package cache are costly to run. Once you know which statements take a long time to run, you can do performance tuning on them.

### Before you begin

The CREATE EVENT MONITOR statement requires a table space with a page size of at least 8 K to store the unformatted event (UE) table produced by the event monitor. Unless a table space is explicitly named in the CREATE EVENT MONITOR statement, the default table space for the database is used.

### About this task

This task shows how you can examine all work done on the system between two points in time to find the costliest statements in terms of total CPU time. Using the package cache event monitor together with package cache information reflected in in-memory monitor elements (as returned by the MON\_GET\_PKG\_CACHE\_STMT or MON\_GET\_PKG\_CACHE\_STMT\_DETAILS table functions) is useful because you can see both statements in the cache as well as statements that have been evicted from the cache. Once the costly statements have been identified, you can then do performance tuning on these statements.

**Note:** You can choose from a number of monitor elements to use when determining which statements are costly to run. In this example, CPU time is used (“total\_cpu\_time - Total CPU time monitor element” on page 1330). This measurement shows actual CPU resources consumed; it does not reflect things like lock wait time or other time spent during statement execution. You might instead choose to use statement execution time (“stmt\_exec\_time - Statement execution time monitor element” on page 1245), which includes the time spent by all agents in the section, and includes wait times, among other things. You can also choose from many of the other time-spent elements returned by the package cache event monitor. See “Information written to relational tables by EVMON\_FORMAT\_UE\_TO\_TABLES for a package cache event monitor” on page 226 or “Information written to XML for a package cache event monitor” on page 235 for more information about which monitor elements you can choose from.

### Restrictions

In this particular example, the length of the analyzed statements is limited to 3000 characters. This limitation is due to the use of the GROUP BY clause used in the statement, which cannot be used with LOB values, such as the **stmt\_text** monitor element.

### Procedure

1. Create a package cache event monitor to capture statements as they are removed (evicted) from the package cache. For example, to create an event monitor called EXPENSIVESTMTS, you could use the following SQL:

```
CREATE EVENT MONITOR EXPENSIVESTMTS FOR PACKAGE CACHE WRITE TO UNFORMATTED EVENT TABLE
```

This statement creates a package cache event monitor that writes to a UE table with the same name as the event monitor, EXPENSIVESTMTS, in the default table space for the database. You can override the default name for the UE table using the TABLE *table-name* clause. You can also override the table space used for the UE table by using the **IN** *tablespace-name* clause.

By default, all statements evicted from the package cache are captured by the package cache event monitor. To limit the amount of information collected, you can specify options as part of the CREATE EVENT MONITOR statement that restrict the information collected. See the documentation for the CREATE EVENT MONITOR (package cache) statement for more information.

- Next, activate the event monitor:

```
SET EVENT MONITOR EXPENSIVESTMTS STATE 1
```

**Note:** By default, this event monitor starts automatically upon database activation, because the AUTOSTART option is applied by default. However, because this event monitor is being created in an already-active database, you must use the **SET EVENT MONITOR** command to start it manually.

- Connect to the database and run whichever statements, workload, or applications for which you are interested in doing performance analysis. You can collect as much information as you like. However, this type of performance tuning works best when you have applications or workloads that run on a regular basis; otherwise adjustments you make for previously executed statements might not have any impact on statements that run in the future.
- When you are finished collecting data, deactivate the event monitor:

```
SET EVENT MONITOR EXPENSIVESTMTS STATE 0
```

- Extract the data from the UE table that was populated by the event monitor using the EVMON\_FORMAT\_UE\_TO\_TABLES procedure.

```
CALL EVMON_FORMAT_UE_TO_TABLES ('PKG_CACHE', NULL, NULL, NULL, NULL, NULL,
NULL, -1, 'SELECT * FROM EXPENSIVESTMTS')
```

This procedure examines the UE table TRACKSTMTS produced by the event monitor. It selects all of the records from the UE table, and from them, creates two relational tables from the data collected by the package cache event monitor:

- PKGCACHE\_EVENT
- PCKCACHE\_METRICS

The first table contains the most frequently used monitor elements and metrics associated with each event captured. The second contains detailed metrics for each event.

**Note:** The values in the columns of PKGCACHE\_METRICS can also be found in the XML document contained in the METRICS column of the PKGCACHE\_EVENT table. They are provided in the PKGCACHE\_METRICS table for more convenient, column-oriented access.

- Query the output from the event monitor to determine which statements took the longest time to run. In this example, total CPU time (“total\_cpu\_time - Total CPU time monitor element” on page 1330) is the time-spent monitor element used to determine overall cost:

```
WITH STMTS AS
(
  1 [ SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
    [ FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-2)) AS T
      [ GROUP BY EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000)
    UNION ALL
    2 [ SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
      [ FROM PKGCACHE_EVENT E, PKGCACHE_METRICS M WHERE E.XMLID = M.XMLID
        [ GROUP BY EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000)
    )
SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, STMT_TEXT, EXECUTABLE_ID
```



command to analyze high-cost statements that were run during that period. The output of the db2advise command suggests adjustments or changes you can make to your database to improve its performance, based on the statements run while the package cache event monitor was active. Using the package cache event monitor to capture these statements is useful if the statements in question are no longer in the package cache.

### Restrictions

In this particular example, the length of the analyzed statements is limited to 3000 characters. This limitation is due to the use of the GROUP BY clause used in the statement, which cannot be used with LOB values, such as the `stmt_text` monitor element.

### Procedure

1. Create a package cache event monitor to capture statements as they are removed (evicted) from the package cache. For example, to create an event monitor called TRACKSTMTS, you could use the following SQL:  

```
CREATE EVENT MONITOR TRACKSTMTS FOR PACKAGE CACHE WRITE TO UNFORMATTED EVENT TABLE
```

This statement creates a package cache event monitor that writes to a UE table with the same name as the event monitor, TRACKSTMTS.
2. Next, activate the event monitor:  

```
SET EVENT MONITOR TRACKSTMTS STATE 1
```
3. Connect to the database and run whichever statements, workload or applications for which you are interested in doing performance analysis. You can collect as much information as you like. However, this type of performance tuning works best when you have applications or workloads that run on a regular basis; otherwise adjustments you make for previously executed statements might not have any impact on statements that run in the future.
4. When you are finished collecting data, deactivate the event monitor:  

```
SET EVENT MONITOR TRACKSTMTS STATE 0
```
5. Extract the data from the UE table that was populated by the event monitor using the EVMON\_FORMAT\_UE\_TO\_TABLES procedure.  

```
CALL EVMON_FORMAT_UE_TO_TABLES  
('PKG_CACHE', NULL, NULL, NULL, NULL, NULL, NULL, -1,  
'SELECT * FROM TRACKSTMTS')
```

This procedure creates two relational tables from the data collected by the package cache event monitor:

- PKG\_CACHE\_EVENT
- PKG\_CACHE\_METRICS

The first table contains the most frequently used monitor elements and metrics associated with each event captured. The second contains detailed metrics for each event.

**Note:** The values in the columns of PKG\_CACHE\_METRICS can also be found in the XML document contained in the METRICS column of the PKG\_CACHE\_EVENT table. They are provided in the PKG\_CACHE\_METRICS table for more convenient, column-oriented access.

6. Query the output from the event monitor to determine which statements took the longest time to run. In this example, statement execution time (“`stmt_exec_time` - Statement execution time monitor element” on page 1245) is

the time-spent monitor element used to determine overall cost. This monitor element is summed across all database partitions.

**Tip:** Save the output from the query into a text file. You will use this file in the next step.

```
WITH STMTS AS
(
  SELECT SUM(TOTAL_STMT_EXEC_TIME)/SUM(TOTAL_NUM_COORD_EXEC_WITH_METRICS) AS AVG_TIME_PER_EXEC,
         STMT_TEXT, SUM(NUM_EXECUTIONS) AS NUM_EXECUTIONS, STMT_TYPE_ID
  FROM (
    (
      SELECT      SUM(STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME,
                  SUM(NUM_COORD_EXEC_WITH_METRICS) AS TOTAL_NUM_COORD_EXEC_WITH_METRICS,
                  SUM(NUM_COORD_EXEC) AS NUM_EXECUTIONS,
                  VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT,
                  STMT_TYPE_ID
      FROM        PKGCACHE_EVENT AS E, PKGCACHE_METRICS AS M
      WHERE       E.XMLID = M.XMLID
      AND         NUM_COORD_EXEC_WITH_METRICS > 0
      GROUP BY   VARCHAR(STMT_TEXT, 3000), STMT_TYPE_ID
      ORDER BY   TOTAL_NUM_COORD_EXEC_WITH_METRICS DESC
      FETCH FIRST 50 ROWS ONLY
    )
    UNION ALL
    (
      SELECT      SUM(STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME,
                  SUM(NUM_COORD_EXEC_WITH_METRICS) AS TOTAL_NUM_COORD_EXEC_WITH_METRICS,
                  SUM(NUM_COORD_EXEC) AS NUM_EXECUTIONS,
                  VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT,
                  STMT_TYPE_ID
      FROM        TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-2)) AS T
      WHERE       NUM_COORD_EXEC_WITH_METRICS > 0
      GROUP BY   VARCHAR(STMT_TEXT, 3000), STMT_TYPE_ID
      ORDER BY   TOTAL_NUM_COORD_EXEC_WITH_METRICS DESC
      FETCH FIRST 50 ROWS ONLY
    )
  ) AS Q_UA
  GROUP BY STMT_TEXT, STMT_TYPE_ID
)
SELECT  '--# SET FREQUENCY ' || NUM_EXECUTIONS || X'0A' || STMT_TEXT || ';'
FROM    STMTS WHERE STMT_TYPE_ID LIKE 'DML, Select%' OR STMT_TYPE_ID LIKE 'DML, Insert%' 1
ORDER BY  AVG_TIME_PER_EXEC DESC
FETCH FIRST 50 ROWS ONLY;
```

In the preceding sample statement, both the data from the package cache event monitor and the in-memory information from the `MON_GET_PKG_CACHE_STMT` table function are retrieved. Looking at both data sets lets you see data for statements evicted from the package cache, as well as statements that still exist in the package cache. Doing so assures that when you evaluate which statements are costly to run that you also include statements not yet evicted from the cache. In each case, the query retrieves the top 50 statements from both the active package cache, and the package cache event monitor, based on the number of times the statements ran. Then, from these statements, the top 50 `SELECT` or `INSERT` statements are chosen **1** based on the average length of time the statements ran for.

**Note:** You can choose from a number of monitor elements to use when determining which statements are costly to run. In this example, statement execution time is used. This measurement includes shows the amount of time spent in execution by all members and agents executing this section, and includes things like wait time. You might instead choose to use CPU time (“total\_cpu\_time - Total CPU time monitor element” on page 1330), which reports only the time spent by the CPU processing the statement. You could also choose from many of the other time-spent elements returned by the package cache event monitor. See “Information written to relational tables by `EVMON_FORMAT_UE_TO_TABLES` for a package cache event monitor” on page 226

page 226 or “Information written to XML for a package cache event monitor” on page 235 for more information about which monitor elements you can choose from.

In addition, the query presents the output in the `--# SET FREQUENCY` format the Design Advisor uses for its analysis. The preceding query returns results like the ones that follow:

```
-----
--# SET FREQUENCY 1
WITH STMTS AS ( SELECT SUM(TOTAL_STMT_EXEC_TIME)/SUM(TOTAL_NUM_COORD_EXEC_WITH_METRICS) AS AVG_TIME_PER_EXEC, STMT
--# SET FREQUENCY 2
WITH STMTS AS ( SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
--# SET FREQUENCY 1055
SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2CommonMED' AND DECISION='NOP' AND NAME='CommonPolicy';
--# SET FREQUENCY 99
SELECT CREATOR, NAME, CTIME FROM SYSIBM.SYSTABLES WHERE TYPE='T' OR TYPE='S' OR TYPE='N' WITH UR;
--# SET FREQUENCY 1
UPDATE SYSTOOLS.HMON_ATM_INFO SET STATS_LOCK = 'N', REORG_LOCK = 'N';
--# SET FREQUENCY 1
UPDATE SYSTOOLS.HMON_ATM_INFO AS ATM SET STATS_FLAG = 'N', REORG_FLAG = 'N' WHERE (ATM.SCHEMA, ATM.NAME) IN (SEL
--# SET FREQUENCY 1
SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2TableMaintenanceMED' AND DECISION='TableRunstatsDecision' AND NAM
--# SET FREQUENCY 83
WITH JTAB(JSHEMA,JNAME) AS (VALUES(TABLE_SCHEMA(CAST(? AS varchar(128))), CAST(? AS varchar(128))), TABLE_NAME (CA
--# SET FREQUENCY 122
WITH VTYPED (NAME, SCHEMA) AS (VALUES(TABLE_NAME (CAST(? AS varchar(128))), CAST(? AS varchar(128))), TABLE_SCHEMA(
--# SET FREQUENCY 1210
SELECT COLNAME, TYPENAME FROM SYSCAT.COLUMNS WHERE TABNAME='POLICY' AND TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 105
SELECT TABNAME FROM SYSCAT.TABLES WHERE TABNAME='HMON_ATM_INFO' AND TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 104
DELETE FROM SYSTOOLS.HMON_ATM_INFO AS ATM WHERE NOT EXISTS ( SELECT * FROM SYSIBM.SYSTABLES AS IBM WHERE ATM.NAME
--# SET FREQUENCY 1118
VALUES(SUBSTR(:H00003 ,:H00014,:H00015 )) INTO :H00009:H00017 ;
--# SET FREQUENCY 274
INSERT INTO "ASRISK"."PKGCACHE_EVENT"("EVENT_ID","XMLID","EVENT_TYPE","EVENT_TIMESTAMP","MEMBER","SECTION_TYPE","I
--# SET FREQUENCY 1
SELECT IBM.TID, IBM.FID FROM SYSIBM.SYSTABLES AS IBM, SYSTOOLS.HMON_ATM_INFO AS ATM WHERE ATM.STATS_FLAG <> 'Y' AN
--# SET FREQUENCY 115
VALUES(SUBSTR(CAST(? AS CLOB(162)),CAST(? AS INTEGER),CAST(? AS INTEGER)));
--# SET FREQUENCY 8227
:
:
--# SET FREQUENCY 532
SELECT TBNAME, TBcreator FROM "ASRISK ".SYSINDEXES WHERE NAME = 'INDCOLUMNS01' AND CREATOR = 'SYSIBM ';
--# SET FREQUENCY 105
SELECT TABNAME FROM SYSCAT.TABLES WHERE TABNAME='HMON_COLLECTION' AND TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 4091
SELECT STATS_LOCK, REORG_LOCK FROM SYSTOOLS.HMON_ATM_INFO WHERE SCHEMA = ? AND NAME = ? AND CREATE_TIME = ? FOR UP
--# SET FREQUENCY 17100
SELECT CREATE_TIME FROM SYSTOOLS.HMON_ATM_INFO WHERE SCHEMA = ? AND NAME = ? FOR UPDATE;
--# SET FREQUENCY 524
SELECT COUNT(*) FROM "SYSIBM".SYSTABLES WHERE NAME = 'SYSDATAPARTITIONEXPRESSION' AND CREATOR = 'SYSIBM ' AND TYP
--# SET FREQUENCY 532
SELECT COUNT(*) FROM "SYSIBM".SYSTABLES WHERE NAME = 'SYSCOLUMNS' AND CREATOR = 'SYSIBM ' AND TYPE = 'S';

47 record(s) selected
```

**Note:** The lines in the preceding sample output have been truncated for presentation purposes.

7. Create an input file for the `db2adviz` command using the statements returned by the query in step 6 on page 249. (For more information about creating input files for the `db2adviz` command, refer to the reference documentation for that command.)
8. Run the `db2adviz` command using the input file created in step 7. For example, if the input file you create is called `pkgcache_stmts.txt`, run a command like the one that follows:

```
db2adviz -d customer -i pkgcache_stmts.txt -m MICP
```

where

- `-d CUSTOMER` identifies the name of the database for which you are getting recommendations
- `-i pkgcache_stmts.txt` identifies the name of the input file for `db2adviz`

- **-m MICP** is a directive to the db2advis command to produce the following recommendations to improve performance:
  - M** New materialized query tables
  - I** New indexes
  - C** Converting standard tables to multidimensional clustering tables (MQTs)
  - P** Repartitioning existing indexes

## Results

The Design Advisor returns recommendations like ones that follow:

```

execution started at timestamp 2010-03-16-14.25.57.562000
Using the default table space name USERSPACE1
found [47] SQL statements from the input file
excluding statement [0] from the workload.
excluding statement [1] from the workload.
excluding statement [19] from the workload.
excluding statement [39] from the workload.
Recommending indexes...
Recommending MQTs...
Recommending Multi-Dimensional Clusterings...
Found 19 user defined views in the catalog table
Found [17] candidate MQTs
Getting cost of workload with MQTs
total disk space needed for initial set [ 0.159] MB
total disk space constrained to [ 69.215] MB
  2 indexes in current solution
  0 MQTs in current solution
total disk space needed for initial set [ 0.024] MB
total disk space constrained to [ 103.822] MB
No useful Multi-dimensional Clustering dimensions for this workload
[5651.8281] timerons (without recommendations)
[5519.8281] timerons (with current solution)
[2.34%] improvement

--
--
-- LIST OF MODIFIED CREATE-TABLE STATEMENTS WITH RECOMMENDED PARTITIONING KEYS AND TABLESPACES AND/OR RECOMMENDED MULTI-DIMENSIONAL CLUSTERINGS
-- =====
-- No new partitioning keys or tablespaces are recommended for this workload.

--
--
-- LIST OF RECOMMENDED MQTs
-- =====

--
--
-- RECOMMENDED EXISTING MQTs
-- =====

--
--
-- UNUSED EXISTING MQTs
-- =====
-- DROP TABLE "ASRISK"."ADEFUSR";

--
--
-- RECOMMENDED CLUSTERING INDEXES
-- =====

--
--
-- LIST OF RECOMMENDED INDEXES
-- =====
-- index[1], 0.024MB
CREATE INDEX "ASRISK"."IDX003161830530000" ON "ASRISK"."SYSINDEXES"
("CREATOR" ASC, "NAME" ASC, "TBCREATOR" ASC, "TBNAME"
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK ;

--
--
-- RECOMMENDED EXISTING INDEXES
-- =====
-- RUNSTATS ON TABLE "SYSTOOLS"."POLICY" FOR SAMPLED DETAILED INDEX "SYSTOOLS"."POLICY_UNQ" ;
-- COMMIT WORK ;

```

```

-- RUNSTATS ON TABLE "SYSTOOLS"."HMON_ATM_INFO" FOR SAMPLED DETAILED INDEX "SYSTOOLS"."ATM_UNIQ" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSDATAPARTITIONS" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDDATAPARTITIONS03" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSTABLES" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDTABLES01" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSTABLESPACES" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDTABLESPACES04" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSCOLUMNS" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDCOLUMNS01" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSINDEXES" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDINDEXES02" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM"."SYSTRIGGERS" FOR SAMPLED DETAILED INDEX "SYSIBM"."INDTRIGGERS02" ;
-- COMMIT WORK ;

--
--
-- UNUSED EXISTING INDEXES
-- =====
-- DROP INDEX "ASRISK"."PKGCACHE_EVENT_IND1";
-- =====
--

-- ===ADVISOR DETAILED XML OUTPUT=====
-- ==(Benefits do not include clustering recommendations)==
:

```

**Note:** The output from the Design Advisor has been truncated for presentation purposes.

### What to do next

Use the output from the Design Advisor to help when deciding what changes to make to your database to improve performance.

## Activity event monitoring

The activity event monitor captures data that is related to activities that run on the system. You can use this event monitor to gather data to help you better understand the performance and behaviour of statements and of the load on your system in general.

The activity event monitor records information after the completion of each activity in the system. By contrast, the unit of work event monitor records data at the completion of each transaction. Using the activity event monitor, you can examine monitor elements related to the execution of individual statements.

The data that the activity event monitor returns complements the data that the following table functions return:

- MON\_GET\_ACTIVITY\_DETAILS
- MON\_GET\_PKG\_CACHE\_STMT
- MON\_GET\_PKG\_CACHE\_STMT\_DETAILS

Whereas the event monitor returns historical information about activities that ran on the system, the table functions provide information about activities that have run or have recently run on the system.

## Uses of the activity event monitor

### Use with other event monitors

The activity event monitor is particularly useful in conjunction with other event monitors. For example, you might want to capture information about the execution of a statement that violates a threshold that you define. In this case, you perform the following steps:

1. Define the threshold by using the CREATE THRESHOLD statement. As part of defining the threshold, specify the COLLECT ACTIVITY DATA clause to have activity data recorded by any active activity event monitors.
2. Create a threshold violations event monitor to capture details about when the threshold was violated, along with other data about what was happening in the system at that time.
3. Create an activity event monitor that captures the activity information that the threshold violation generates.
4. Run your application or workload.
5. Query the event monitor output to view information about what was happening when the threshold was violated. You might perform a join of data from the threshold violations event monitor with the data from the activity event monitor to pinpoint the statement that was running when the violation occurred.

This process is illustrated in more detail in “Example: Capturing activity information related to the execution of a statement” on page 270.

Other applications of the activity event monitor include the following ones:

- Capturing information about long-running queries. In this case, you run the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS procedure to force the collection of information about an activity before the activity is finished. Running this procedure is useful if you want to terminate a long-running statement but also want to capture information about it.
- Seeing what statements the applications in a specific workload are running.

#### **Input to commands, procedures, or tools**

The data produced by the activity event monitor can be used as input for various tools and stored procedures, including the ones that follow:

##### **db2advls - DB2 Design Advisor command**

The **db2advls** command can use the output of the activity event monitor to produce recommendations about the following items and activities:

- Materialized query tables (MQTs)
- Indexes
- Repartitioning of tables
- Conversion to multidimensional clustering (MDC) tables
- Deletion of unused objects

##### **db2expln - SQL and XQuery Explain command**

The **db2expln** command can use section information from the activity event monitor to describe the access plan for statements related to a section.

##### **EXPLAIN\_FROM\_ACTIVITY stored procedure**

The EXPLAIN\_FROM\_ACTIVITY stored procedure explains a specific execution of a statement by using the contents of the section that the procedure obtains from an activity event monitor. The Explain output is placed in the Explain tables for processing using Explain tools (for example, the **db2exfmt** command). The Explain output contains, if available, both the access plan and section actuals (runtime statistics for operators in the access plan).

### Workload management historical analysis tools

The `wlmhist.pl` and `wlmhistrep.pl` Perl scripts perform historical analysis by using information that the activities event monitor captures.

### Section actuals

Another use of the activity event monitor is to capture section actuals. You can use this data to compare the actual values that an activity event monitor captures with the estimated costs in an access plan. Doing this comparison can let you see whether the access plan is still valid.

## Creating an activity event monitor

To create an event monitor that captures activity events, use the `CREATE EVENT MONITOR FOR ACTIVITIES` statement.

### Before you begin

You must have either `DBADM` or `SQLADM` authority to create an activity event monitor.

### Procedure

To create an activities event monitor:

1. Formulate a `CREATE EVENT MONITOR FOR ACTIVITIES` statement. For example, to create an event monitor called `myactevmon`, you might use a statement such as the one that follows:

```
CREATE EVENT MONITOR myactevmon FOR ACTIVITIES
WRITE TO TABLE
```

2. Run the statement. This example creates an activities event monitor called `myactevmon` with the following characteristics:
  - Monitor elements for all logical data groups that are applicable to activity event monitors are collected.
  - Output is written to relational tables. Default table names are assigned to each table.
  - The event monitor is configured to start automatically when the database is first activated.
  -

These results are due to using the default settings for activity event monitors. You can override the defaults if necessary.

3. Activate the event monitor. Although the event monitor is configured to start automatically, it will not do so until the first database activation subsequent to the creation of the event monitor. To force the event monitor to start collecting data immediately, use the `SET EVENT MONITOR STATE` statement, as shown in the following example:

```
SET EVENT MONITOR myactevmon STATE 1
```

### What to do next

Configure the event monitor to collect the data that you want.

## Configuring data collection for an activity event monitor

Before you can collect event data related to activities, you must configure data collection. The configuration options that you can use depend on your purpose for collecting data.

## About this task

There are two ways to configure data collection for an activity event monitor:

- You can have activity event data generated whenever a threshold that you created by using the CREATE THRESHOLD statement is violated.
- You can specify the COLLECT ACTIVITY METRICS clause for the CREATE or ALTER statement for WLM objects.

## Procedure

To collect event data that is related to activities:

1. Determine the activities for which you want to collect data. The table that follows outlines the options:

| For which activities is data to be collected?      | Configuration controlled by:  |
|--|---|
| <b>Activities related to a threshold violation</b> | For activities that are related to a threshold violation, issue the CREATE THRESHOLD or ALTER THRESHOLD statement, specifying the COLLECT ACTIVITY DATA clause. When you specify the COLLECT ACTIVITY DATA clause, activity data is sent to any active activities event monitors whenever the threshold is violated. Optionally, specify one of the following additional clauses: WITH DETAILS; WITH DETAILS, SECTION; or AND VALUES.<br><b>Tip:</b> Consider changing the value of the <b>mon_act_metrics</b> configuration parameter to NONE. Doing so restricts the activities for which event data that is collected to just those activities that are related to threshold violations. |

| For which activities is data to be collected?                            | Configuration controlled by:   |
|--|--|
| <p><b>Activities related to specific workload management objects</b></p> | <p>For activities related to workload management, issue one of the following statements, specifying the COLLECT ACTIVITY DATA clause. Optionally, specify one of the following additional clauses: WITH DETAILS; WITH DETAILS, SECTION; WITH SECTION INCLUDE ACTUALS BASE; or AND VALUES:</p> <ul style="list-style-type: none"> <li>• CREATE WORKLOAD</li> <li>• ALTER WORKLOAD</li> <li>• CREATE WORK ACTION SET</li> <li>• ALTER WORK ACTION SET</li> <li>• CREATE SERVICE CLASS</li> <li>• ALTER SERVICE CLASS</li> </ul> <p>When you specify a COLLECT ACTIVITY DATA clause for any of these statements, activity data for the workload objects that are referenced in the statement is sent to any active activity event monitor.</p> <p>For example, to collect activity event data for the PAYROLL workload, you can alter the workload by issuing this statement:</p> <pre>ALTER WORKLOAD PAYROLL COLLECT ACTIVITY DATA WITH SECTION INCLUDE ACTUALS BASE</pre> <p>In this example, activity data is collected along with data for section actuals.</p> |

- Set the collection level using one of the configuration options described in the previous step. For example, to collect activity event data for the PAYROLL workload, you can alter the workload with this statement:

```
ALTER WORKLOAD PAYROLL COLLECT ACTIVITY DATA WITH SECTION INCLUDE ACTUALS BASE
```

In this last example, activity data is collected, along with data for section actuals.

## Results

Data collection is configured.

### Collecting activity data for the default user workload

If you want to collect activity event data for activities that do not map to a specific workload, you can collect data for the default user workload by issuing a statement such as the one that follows:

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS
```

This statement causes detailed activity data to be collected for any activity that is not associated with a user-defined workload object. By default, all database connections are associated with the SYSDEFAULTUSERWORKLOAD workload if there are no user-defined workload objects.

## What to do next

Run your application or workload. As activities that satisfy the collection criteria are run, event data is sent to any active activity event monitors.

## Data generated by activities event monitors

You can choose to have the output from a activities event monitor written to regular tables, files or named pipes.

Regardless of the output format you choose, all data captured by activities event monitors comes from one of four logical data groups:

- event\_activity
- event\_activitymetrics
- event\_activitystmt
- event\_activityvals

If you choose to have the statistics event monitor data written to regular tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for an activities event monitor:

Information written by an activities event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for an activities event monitor, by default, five tables are produced, each containing monitor elements from one or more logical data groups:

*Table 47. Tables produced by ACTIVITIES write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name                 | Logical data groups reported  |
|------------------------------------|---|
| ACTIVITY_ <i>evmon-name</i>        | event_activity  |
| ACTIVITYSTMT_ <i>evmon-name</i>    | event_activitystmt  |
| ACTIVITYVALS_ <i>evmon-name</i>    | event_activityvals  |
| ACTIVITYMETRICS_ <i>evmon-name</i> | event_activitymetrics   |
| CONTROL_ <i>evmon-name</i>         | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

## Tables produced

Table 48. Information returned for a activities event monitor: Default table name: ACTIVITY\_evmon-name

| Column Name             | Data Type    | Description  |
|-------------------------|--------------|--|
| PARTITION_KEY           | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016      |
| ACT_EXEC_TIME           | BIGINT       | act_exec_time - Activity execution time                              |
| ACTIVATE_TIMESTAMP      | TIMESTAMP    | activate_timestamp - Activate timestamp                              |
| ACTIVITY_ID             | BIGINT       | activity_id - Activity ID  |
| ACTIVITY_SECONDARY_ID   | SMALLINT     | activity_secondary_id - Activity secondary ID                        |
| ACTIVITY_TYPE           | VARCHAR(64)  | activity_type - Activity type  |
| ADDRESS                 | VARCHAR(128) | address - IP address from which the connection was initiated         |
| AGENT_ID                | BIGINT       | agent_id - Application handle (agent ID)                             |
| APPL_ID                 | VARCHAR(64)  | appl_id - Application ID   |
| APPL_NAME               | VARCHAR(255) | appl_name - Application name   |
| ARM_CORRELATOR          | BLOB(0)      | arm_correlator - Application response measurement correlator         |
| COORD_PARTITION_NUM     | INTEGER      | coord_partition_num - Coordinator partition number                   |
| DB_WORK_ACTION_SET_ID   | INTEGER      | db_work_action_set_id - Database work action set ID                  |
| DB_WORK_CLASS_ID        | INTEGER      | db_work_class_id - Database work class ID                            |
| DETAILS_XML             | BLOB(0)      |  |
| MON_INTERVAL_ID         | BIGINT       | mon_interval_id - Monitor interval identifier                        |
| NUM_REMAPS              | BIGINT       | num_remaps - Number of remaps  |
| PARENT_ACTIVITY_ID      | BIGINT       | parent_activity_id - Parent activity ID                              |
| PARENT_UOW_ID           | INTEGER      | parent_uow_id - Parent unit of work ID                               |
| PARTIAL_RECORD          | SMALLINT     | partial_record - Partial record                                      |
| PARTITION_NUMBER        | SMALLINT     | partition_number - Partition number                                  |
| POOL_DATA_L_READS       | BIGINT       | pool_data_l_reads - Buffer pool data logical reads                   |
| POOL_DATA_P_READS       | BIGINT       | pool_data_p_reads - Buffer pool data physical reads                  |
| POOL_INDEX_L_READS      | BIGINT       | pool_index_l_reads - Buffer pool index logical reads                 |
| POOL_INDEX_P_READS      | BIGINT       | pool_index_p_reads - Buffer pool index physical reads                |
| POOL_TEMP_DATA_L_READS  | BIGINT       | pool_temp_data_l_reads - Buffer pool temporary data logical reads    |
| POOL_TEMP_DATA_P_READS  | BIGINT       | pool_temp_data_p_reads - Buffer pool temporary data physical reads   |
| POOL_TEMP_INDEX_L_READS | BIGINT       | pool_temp_index_l_reads - Buffer pool temporary index logical reads  |
| POOL_TEMP_INDEX_P_READS | BIGINT       | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS   | BIGINT       | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |

Table 48. Information returned for a activities event monitor: Default table name: ACTIVITY\_evmon-name (continued)

| Column Name             | Data Type    | Description   |
|-------------------------|--------------|---|
| POOL_TEMP_XDA_P_READS   | BIGINT       | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads     |
| POOL_XDA_L_READS        | BIGINT       | pool_xda_l_reads - Buffer pool XDA data logical reads                     |
| POOL_XDA_P_READS        | BIGINT       | pool_xda_p_reads - Buffer pool XDA data physical reads                    |
| PREP_TIME               | BIGINT       | prep_time - Preparation time  |
| QUERY_ACTUAL_DEGREE     | INTEGER      | query_actual_degree - Actual runtime degree of intrapartition parallelism |
| QUERY_CARD_ESTIMATE     | BIGINT       | query_card_estimate - Query number of rows estimate                       |
| QUERY_COST_ESTIMATE     | BIGINT       | query_cost_estimate - Query cost estimate                                 |
| QUERY_DATA_TAG_LIST     | VARCHAR(32)  | query_data_tag_list - Query data tag list                                 |
| ROWS_FETCHED            | BIGINT       | rows_fetched - Rows fetched   |
| ROWS_MODIFIED           | BIGINT       | rows_modified - Rows modified   |
| ROWS_RETURNED           | BIGINT       | rows_returned - Rows returned   |
| SC_WORK_ACTION_SET_ID   | INTEGER      | sc_work_action_set_id - Service class work action set ID                  |
| SC_WORK_CLASS_ID        | INTEGER      | sc_work_class_id - Service class work class ID                            |
| SECTION_ACTUALS         | BLOB(0)      | section_actuals - Section actuals   |
| SERVICE_SUBCLASS_NAME   | VARCHAR(128) | service_subclass_name - Service subclass name                             |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name                         |
| SESSION_AUTH_ID         | VARCHAR(128) | session_auth_id - Session authorization ID                                |
| SORT_OVERFLOWS          | BIGINT       | sort_overflows - Sort overflows   |
| SQLCABC                 | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLCAID                 | CHARACTER(8) | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLCODE                 | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD1                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD2                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD3                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD4                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD5                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRD6                | INTEGER      | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |
| SQLERRM                 | VARCHAR(72)  | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .  |

Table 48. Information returned for a activities event monitor: Default table name: ACTIVITY\_evmon-name (continued)

| Column Name                  | Data Type     | Description  |
|------------------------------|---------------|--|
| SQLERRP                      | CHARACTER(8)  | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLSTATE                     | CHARACTER(5)  | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLWARN                      | CHARACTER(11) | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SYSTEM_CPU_TIME              | BIGINT        | system_cpu_time - System CPU time  |
| TIME_COMPLETED               | TIMESTAMP     | time_completed - Time completed  |
| TIME_CREATED                 | TIMESTAMP     | time_created - Time created  |
| TIME_STARTED                 | TIMESTAMP     | time_started - Time started  |
| TOTAL_SORT_TIME              | BIGINT        | total_sort_time - Total sort time  |
| TOTAL_SORTS                  | BIGINT        | total_sorts - Total sorts  |
| TOTAL_STATS_FABRICATION_TIME | BIGINT        | total_stats_fabrication_time - Total statistics fabrication time         |
| TOTAL_STATS_FABRICATIONS     | BIGINT        | total_stats_fabrications - Total statistics fabrications                 |
| TOTAL_SYNC_RUNSTATS          | BIGINT        | total_sync_runstats - Total synchronous RUNSTATS activities              |
| TOTAL_SYNC_RUNSTATS_TIME     | BIGINT        | total_sync_runstats_time - Total synchronous RUNSTATS time               |
| TPMON_ACC_STR                | VARCHAR(200)  | tpmon_acc_str - TP monitor client accounting string                      |
| TPMON_CLIENT_APP             | VARCHAR(255)  | tpmon_client_app - TP monitor client application name                    |
| TPMON_CLIENT_USERID          | VARCHAR(255)  | tpmon_client_userid - TP monitor client user ID                          |
| TPMON_CLIENT_WKSTN           | VARCHAR(255)  | tpmon_client_wkstn - TP monitor client workstation name                  |
| UOW_ID                       | INTEGER       | uow_id - Unit of work ID   |
| USER_CPU_TIME                | BIGINT        | user_cpu_time - User CPU time  |
| WL_WORK_ACTION_SET_ID        | INTEGER       | wl_work_action_set_id - Workload work action set identifier              |
| WL_WORK_CLASS_ID             | INTEGER       | wl_work_class_id - Workload work class identifier                        |
| WORKLOAD_ID                  | INTEGER       | workload_id - Workload ID  |
| WORKLOAD_OCCURRENCE_ID       | INTEGER       | workload_occurrence_id - Workload occurrence identifier                  |

Table 49. Information returned for an activities event monitor: Table name: ACTIVITYSTMT\_evmon-name

| Column Name           | Data Type   | Description   |
|-----------------------|-------------|---|
| PARTITION_KEY         | INTEGER     | "partition_key - Partitioning key monitor element" on page 1016 |
| ACTIVATE_TIMESTAMP    | TIMESTAMP   | activate_timestamp - Activate timestamp                         |
| ACTIVITY_ID           | BIGINT      | activity_id - Activity ID                                       |
| ACTIVITY_SECONDARY_ID | SMALLINT    | activity_secondary_id - Activity secondary ID                   |
| APPL_ID               | VARCHAR(64) | appl_id - Application ID  |

Table 49. Information returned for an activities event monitor: Table name: ACTIVITYSTMT\_evmon-name (continued)

| Column Name         | Data Type    | Description   |
|---------------------|--------------|---|
| COMP_ENV_DESC       | BLOB(0)      | comp_env_desc - Compilation environment               |
| CREATOR             | VARCHAR(128) | creator - Application creator                         |
| EFF_STMT_TEXT       | CLOB         | eff_stmt_text - Effective statement text              |
| EXECUTABLE_ID       | VARCHAR(32)  | executable_id - Executable ID                         |
| PACKAGE_NAME        | VARCHAR(128) | package_name - Package name                           |
| PACKAGE_VERSION_ID  | VARCHAR(64)  | package_version_id - Package version                  |
| PARTITION_NUMBER    | SMALLINT     | partition_number - Partition number                   |
| ROUTINE_ID          | BIGINT       | routine_id - Routine ID                               |
| SECTION_ENV         | BLOB(0)      | section_env - Section environment                     |
| SECTION_NUMBER      | BIGINT       | section_number - Section number                       |
| STMT_FIRST_USE_TIME | TIMESTAMP    | stmt_first_use_time - Statement first use timestamp   |
| STMT_INVOCATION_ID  | BIGINT       | stmt_invocation_id - Statement invocation identifier  |
| STMT_ISOLATION      | BIGINT       | stmt_isolation - Statement isolation                  |
| STMT_LAST_USE_TIME  | TIMESTAMP    | stmt_last_use_time - Statement last use timestamp     |
| STMT_LOCK_TIMEOUT   | INTEGER      | stmt_lock_timeout - Statement lock timeout            |
| STMT_NEST_LEVEL     | BIGINT       | stmt_nest_level - Statement nesting level             |
| STMT_PKG_CACHE_ID   | BIGINT       | stmt_pkgcache_id - Statement package cache identifier |
| STMT_QUERY_ID       | BIGINT       | stmt_query_id - Statement query identifier            |
| STMT_SOURCE_ID      | BIGINT       | stmt_source_id - Statement source identifier          |
| STMT_TEXT           | CLOB         | stmt_text - SQL statement text                        |
| STMT_TYPE           | BIGINT       | stmt_type - Statement type                            |
| UOW_ID              | INTEGER      | uow_id - Unit of work ID                              |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name

| Column Name                 | Data Type   | Description  |
|-----------------------------|-------------|--|
| PARTITION_KEY               | INTEGER     | "partition_key - Partitioning key monitor element" on page 1016        |
| ACTIVITY_ID                 | BIGINT      | activity_id - Activity ID  |
| ACTIVITY_SECONDARY_ID       | SMALLINT    | activity_secondary_id - Activity secondary ID                          |
| APPL_ID                     | VARCHAR(64) | appl_id - Application ID   |
| PARTITION_NUMBER            | SMALLINT    | partition_number - Partition number                                    |
| UOW_ID                      | INTEGER     | uow_id - Unit of work ID   |
| WLM_QUEUE_TIME_TOTAL        | BIGINT      | wlm_queue_time_total - Workload manager total queue time               |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT      | wlm_queue_assignments_total - Workload manager total queue assignments |
| FCM_TQ_RECV_WAIT_TIME       | BIGINT      | fcm_tq_recv_wait_time - FCM table queue received wait time             |
| FCM_MESSAGE_RECV_WAIT_TIME  | BIGINT      | fcm_message_recv_wait_time - FCM message received wait time            |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name                  | Data Type | Description   |
|------------------------------|-----------|---|
| FCM_TQ_SEND_WAIT_TIME        | BIGINT    | fcm_tq_send_wait_time - FCM table queue send wait time            |
| FCM_MESSAGE_SEND_WAIT_TIME   | BIGINT    | fcm_message_send_wait_time - FCM message send wait time           |
| LOCK_WAIT_TIME               | BIGINT    | lock_wait_time - Time waited on locks                             |
| LOCK_WAITS                   | BIGINT    | lock_waits - Lock waits   |
| DIRECT_READ_TIME             | BIGINT    | direct_read_time - Direct read time                               |
| DIRECT_READ_REQS             | BIGINT    | direct_read_reqs - Direct read requests                           |
| DIRECT_WRITE_TIME            | BIGINT    | direct_write_time - Direct write time                             |
| DIRECT_WRITE_REQS            | BIGINT    | direct_write_reqs - Direct write requests                         |
| LOG_BUFFER_WAIT_TIME         | BIGINT    | log_buffer_wait_time - Log buffer wait time                       |
| NUM_LOG_BUFFER_FULL          | BIGINT    | num_log_buffer_full - Number of full log buffers                  |
| LOG_DISK_WAIT_TIME           | BIGINT    | log_disk_wait_time - Log disk wait time                           |
| LOG_DISK_WAITS_TOTAL         | BIGINT    | log_disk_waits_total - Total log disk waits                       |
| POOL_WRITE_TIME              | BIGINT    | pool_write_time - Total buffer pool physical write time           |
| POOL_READ_TIME               | BIGINT    | pool_read_time - Total buffer pool physical read time             |
| AUDIT_FILE_WRITE_WAIT_TIME   | BIGINT    | audit_file_write_wait_time - Audit file write wait time           |
| AUDIT_FILE_WRITES_TOTAL      | BIGINT    | audit_file_writes_total - Total audit files written               |
| AUDIT_SUBSYSTEM_WAIT_TIME    | BIGINT    | audit_subsystem_wait_time - Audit subsystem wait time             |
| AUDIT_SUBSYSTEM_WAITS_TOTAL  | BIGINT    | audit_subsystem_waits_total - Total audit subsystem waits         |
| DIAGLOG_WRITE_WAIT_TIME      | BIGINT    | diaglog_write_wait_time - Diagnostic log file write wait time     |
| DIAGLOG_WRITES_TOTAL         | BIGINT    | diaglog_writes_total - Total diagnostic log file writes           |
| FCM_SEND_WAIT_TIME           | BIGINT    | fcm_send_wait_time - FCM send wait time                           |
| FCM_RECV_WAIT_TIME           | BIGINT    | fcm_recv_wait_time - FCM received wait time                       |
| TOTAL_ACT_WAIT_TIME          | BIGINT    | total_act_wait_time - Total activity wait time                    |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT    | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS          | BIGINT    | total_section_sorts - Total section sorts                         |
| TOTAL_SECTION_SORT_TIME      | BIGINT    | total_section_sort_time - Total section sort time                 |
| TOTAL_ACT_TIME               | BIGINT    | total_act_time - Total activity time                              |
| ROWS_READ                    | BIGINT    | rows_read - Rows read   |
| ROWS_MODIFIED                | BIGINT    | rows_modified - Rows modified                                     |
| POOL_DATA_L_READS            | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                |
| POOL_INDEX_L_READS           | BIGINT    | pool_index_l_reads - Buffer pool index logical reads              |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name             | Data Type | Description   |
|-------------------------|-----------|---|
| POOL_TEMP_DATA_L_READS  | BIGINT    | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_INDEX_L_READS | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_XDA_L_READS        | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_TEMP_XDA_L_READS   | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| TOTAL_CPU_TIME          | BIGINT    | total_cpu_time - Total CPU time                                       |
| POOL_DATA_P_READS       | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_TEMP_DATA_P_READS  | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_XDA_P_READS        | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_TEMP_XDA_P_READS   | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS      | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_TEMP_INDEX_P_READS | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_DATA_WRITES        | BIGINT    | pool_data_writes - Buffer pool data writes                            |
| POOL_XDA_WRITES         | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| POOL_INDEX_WRITES       | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| DIRECT_READS            | BIGINT    | direct_reads - Direct reads from database                             |
| DIRECT_WRITES           | BIGINT    | direct_writes - Direct writes to database                             |
| ROWS_RETURNED           | BIGINT    | rows_returned - Rows returned   |
| DEADLOCKS               | BIGINT    | deadlocks - Deadlocks detected  |
| LOCK_TIMEOUTS           | BIGINT    | lock_timeouts - Number of lock timeouts                               |
| LOCK_ESCALS             | BIGINT    | lock_escals - Number of lock escalations                              |
| FCM_SENDS_TOTAL         | BIGINT    | fcm_sends_total - FCM sends total                                     |
| FCM_RECVS_TOTAL         | BIGINT    | fcm_recvs_total - FCM receives total                                  |
| FCM_SEND_VOLUME         | BIGINT    | fcm_send_volume - FCM send volume                                     |
| FCM_RECV_VOLUME         | BIGINT    | fcm_recv_volume - FCM received volume                                 |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT    | fcm_message_sends_total - Total FCM message sends                     |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT    | fcm_message_recvs_total - Total FCM message receives                  |
| FCM_MESSAGE_SEND_VOLUME | BIGINT    | fcm_message_send_volume - FCM message send volume                     |
| FCM_MESSAGE_RECV_VOLUME | BIGINT    | fcm_message_recv_volume - FCM message received volume                 |
| FCM_TQ_SENDS_TOTAL      | BIGINT    | fcm_tq_sends_total - FCM table queue send total                       |
| FCM_TQ_RECVS_TOTAL      | BIGINT    | fcm_tq_recvs_total - FCM table queue receives total                   |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name                       | Data Type | Description   |
|-----------------------------------|-----------|---|
| FCM_TQ_SEND_VOLUME                | BIGINT    | fcm_tq_send_volume - FCM table queue send volume                            |
| FCM_TQ_RECV_VOLUME                | BIGINT    | fcm_tq_recv_volume - FCM table queue received volume                        |
| TQ_TOT_SEND_SPILLS                | BIGINT    | tq_tot_send_spills - Total number of table queue buffers overflowed         |
| POST_THRESHOLD_SORTS              | BIGINT    | post_threshold_sorts - Post threshold sorts                                 |
| POST_SHRTHRESHOLD_SORTS           | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts                       |
| SORT_OVERFLOWS                    | BIGINT    | sort_overflows - Sort overflows   |
| AUDIT_EVENTS_TOTAL                | BIGINT    | audit_events_total - Total audit events                                     |
| TOTAL_SORTS                       | BIGINT    | total_sorts - Total sorts   |
| STMT_EXEC_TIME                    | BIGINT    | stmt_exec_time - Statement execution time                                   |
| COORD_STMT_EXEC_TIME              | BIGINT    | coord_stmt_exec_time - Execution time for statement by coordinator agent    |
| TOTAL_ROUTINE_NON_SECT_PROC_TIME  | BIGINT    | total_routine_non_sect_proc_time - Non-section processing time              |
| TOTAL_ROUTINE_NON_SECT_TIME       | BIGINT    | total_routine_non_sect_time - Non-section routine execution time            |
| TOTAL_SECTION_PROC_TIME           | BIGINT    | total_section_proc_time - Total section processing time                     |
| TOTAL_APP_SECTION_EXECUTIONS      | BIGINT    | total_app_section_executions - Total application section executions         |
| TOTAL_SECTION_TIME                | BIGINT    | total_section_time - Total section time                                     |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT    | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME      | BIGINT    | total_routine_user_code_time - Total routine user code time                 |
| TOTAL_ROUTINE_TIME                | BIGINT    | total_routine_time - Total routine time                                     |
| THRESH_VIOLATIONS                 | BIGINT    | thresh_violations - Number of threshold violations                          |
| NUM_LW_THRESH_EXCEEDED            | BIGINT    | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded            |
| TOTAL_ROUTINE_INVOCATIONS         | BIGINT    | total_routine_invocations - Total routine invocations                       |
| LOCK_WAIT_TIME_GLOBAL             | BIGINT    | lock_wait_time_global - Lock wait time global                               |
| LOCK_WAITS_GLOBAL                 | BIGINT    | lock_waits_global - Lock waits global                                       |
| RECLAIM_WAIT_TIME                 | BIGINT    | reclaim_wait_time - Reclaim wait time                                       |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME    | BIGINT    | spacemappage_reclaim_wait_time - Space map page reclaim wait time           |
| LOCK_TIMEOUTS_GLOBAL              | BIGINT    | lock_timeouts_global - Lock timeouts global                                 |
| LOCK_ESCALS_MAXLOCKS              | BIGINT    | lock_escals_maxlocks - Number of maxlocks lock escalations                  |
| LOCK_ESCALS_LOCKLIST              | BIGINT    | lock_escals_locklist - Number of locklist lock escalations                  |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name                      | Data Type | Description  |
|----------------------------------|-----------|--|
| LOCK_ESCALS_GLOBAL               | BIGINT    | lock_escals_global - Number of global lock escalations                               |
| CF_WAIT_TIME                     | BIGINT    | cf_wait_time - cluster caching facility wait time                                    |
| CF_WAITS                         | BIGINT    | cf_waits - Number of cluster caching facility DB2 pureScale server waits             |
| POOL_DATA_GBP_L_READS            | BIGINT    | pool_data_gbp_l_reads - Group buffer pool data logical reads                         |
| POOL_DATA_GBP_P_READS            | BIGINT    | pool_data_gbp_p_reads - Group buffer pool data physical reads                        |
| POOL_DATA_LBP_PAGES_FOUND        | BIGINT    | pool_data_lbp_pages_found - Local buffer pool found data pages                       |
| POOL_DATA_GBP_INVALID_PAGES      | BIGINT    | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages                   |
| POOL_INDEX_GBP_L_READS           | BIGINT    | pool_index_gbp_l_reads - Group buffer pool index logical reads                       |
| POOL_INDEX_GBP_P_READS           | BIGINT    | pool_index_gbp_p_reads - Group buffer pool index physical reads                      |
| POOL_INDEX_LBP_PAGES_FOUND       | BIGINT    | pool_index_lbp_pages_found - Local buffer pool index pages found                     |
| POOL_INDEX_GBP_INVALID_PAGES     | BIGINT    | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages                 |
| POOL_XDA_GBP_L_READS             | BIGINT    | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests              |
| POOL_XDA_GBP_P_READS             | BIGINT    | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests             |
| POOL_XDA_LBP_PAGES_FOUND         | BIGINT    | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found                    |
| POOL_XDA_GBP_INVALID_PAGES       | BIGINT    | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages                |
| EVMON_WAIT_TIME                  | BIGINT    | evmon_wait_time - Event monitor wait time  |
| EVMON_WAITS_TOTAL                | BIGINT    | evmon_waits_total - Event monitor total waits  |
| TOTAL_EXTENDED_LATCH_WAIT_TIME   | BIGINT    | total_extended_latch_wait_time - Total extended latch wait time                      |
| TOTAL_EXTENDED_LATCH_WAITS       | BIGINT    | total_extended_latch_waits - Total extended latch waits                              |
| TOTAL_DISP_RUN_QUEUE_TIME        | BIGINT    | total_disp_run_queue_time - Total dispatcher run queue time                          |
| POOL_QUEUED_ASYNC_DATA_REQS      | BIGINT    | pool_queued_async_data_reqs - Data prefetch requests                                 |
| POOL_QUEUED_ASYNC_INDEX_REQS     | BIGINT    | pool_queued_async_index_reqs - Index prefetch requests                               |
| POOL_QUEUED_ASYNC_XDA_REQS       | BIGINT    | pool_queued_async_xda_reqs - XDA prefetch requests                                   |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT    | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name                        | Data Type | Description  |
|------------------------------------|-----------|--|
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces         |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces        |
| POOL_QUEUED_ASYNC_OTHER_REQS       | BIGINT    | pool_queued_async_other_reqs - Non-prefetch requests   |
| POOL_QUEUED_ASYNC_DATA_PAGES       | BIGINT    | pool_queued_async_data_pages - Data pages prefetch requests                                    |
| POOL_QUEUED_ASYNC_INDEX_PAGES      | BIGINT    | pool_queued_async_index_pages - Index pages prefetch requests                                  |
| POOL_QUEUED_ASYNC_XDA_PAGES        | BIGINT    | pool_queued_async_xda_pages - XDA pages prefetch requests                                      |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES  | BIGINT    | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces    |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT    | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces  |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES   | BIGINT    | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS        | BIGINT    | pool_failed_async_data_reqs - Failed data prefetch requests                                    |
| POOL_FAILED_ASYNC_INDEX_REQS       | BIGINT    | pool_failed_async_index_reqs - Failed index prefetch requests                                  |
| POOL_FAILED_ASYNC_XDA_REQS         | BIGINT    | pool_failed_async_xda_reqs - Failed XDA prefetch requests                                      |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS   | BIGINT    | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces    |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS  | BIGINT    | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces  |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS    | BIGINT    | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces      |
| POOL_FAILED_ASYNC_OTHER_REQS       | BIGINT    | pool_failed_async_other_reqs - Failed non-prefetch requests                                    |
| TOTAL_PEDS                         | BIGINT    | total_peds - Total partial early distincts   |
| DISABLED_PEDS                      | BIGINT    | “disabled_peds - Disabled partial early distincts monitor element” on page 802                 |
| POST_THRESHOLD_PEDS                | BIGINT    | post_threshold_peds - Partial early distincts threshold  |
| TOTAL_PEAS                         | BIGINT    | total_peas - Total partial early aggregations  |
| POST_THRESHOLD_PEAS                | BIGINT    | post_threshold_peas - Partial early aggregation threshold                                      |
| TQ_SORT_HEAP_REQUESTS              | BIGINT    | tq_sort_heap_requests - Table queue sort heap requests   |
| TQ_SORT_HEAP_REJECTIONS            | BIGINT    | tq_sort_heap_rejections - Table queue sort heap rejections                                     |
| PREFETCH_WAIT_TIME                 | BIGINT    | prefetch_wait_time - Time waited for prefetch  |
| PREFETCH_WAITS                     | BIGINT    | prefetch_waits - Prefetcher wait count   |

Table 50. Information returned for an activities event monitor: Table name: ACTIVITYMETRICS\_evmon-name (continued)

| Column Name                             | Data Type | Description   |
|---|-----------|---|
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP  | BIGINT    | "pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element" on page 1041   |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT    | "pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element" on page 1073 |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP   | BIGINT    | "pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element" on page 1130     |

Table 51. Information returned for an activities event monitor: Table name: ACTIVITYVALS\_evmon-name

| Column Name           | Data Type     | Description   |
|-----------------------|---------------|---|
| PARTITION_KEY         | INTEGER       | "partition_key - Partitioning key monitor element" on page 1016 |
| ACTIVATE_TIMESTAMP    | TIMESTAMP     | activate_timestamp - Activate timestamp                         |
| ACTIVITY_ID           | BIGINT        | activity_id - Activity ID                                       |
| ACTIVITY_SECONDARY_ID | SMALLINT      | activity_secondary_id - Activity secondary ID                   |
| APPL_ID               | VARCHAR(64)   | appl_id - Application ID  |
| PARTITION_NUMBER      | SMALLINT      | partition_number - Partition number                             |
| STMT_VALUE_DATA       | CLOB          | stmt_value_data - Value data                                    |
| STMT_VALUE_INDEX      | INTEGER       | stmt_value_index - Value index                                  |
| STMT_VALUE_ISNULL     | INTEGER       | stmt_value_isnull - Value has null value                        |
| STMT_VALUE_ISREOPT    | INTEGER       | stmt_value_isreopt - Variable used for statement reoptimization |
| STMT_VALUE_TYPE       | CHARACTER(16) | stmt_value_type - Value type                                    |
| UOW_ID                | INTEGER       | uow_id - Unit of work ID  |

Table 52. Information returned for an activities event monitor: Default table name: CONTROL\_evmon-name

| Column Name        | Data Type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

## Accessing event information written by an activity event monitor to a table

Activity event monitors can write their output to tables, files, and pipes.

For more information about using data written to files and pipes, see “Event monitor self-describing data stream” on page 108.

## Before you begin

You must have created and activated an activity event monitor and enabled data collection.

## About this task

The tables that an activity event monitor produces are described in “Target tables, control tables, and event monitor table management” on page 88. Before DB2 Version 10.1, metrics monitor elements were written to an XML document in the `DETAIL_XML` column of the `ACTIVITIES` table. The schema for the XML document is in the `sql1lib/misc/DB2MonCommon.xsd` file, and the top-level element is `activity_metrics`. Starting in Version 10.1, the metrics formerly available only from the `details_xml` XML document are also available in an `ACTIVITYMETRICS` table that the activity event monitor produces.

## Procedure

To access the data produced by an activity event monitor:

1. Formulate a query that returns the columns that you want to see. For example, if you are interested in information about the text of the statements that are associated with a specific unit of work, you might formulate a query such as the one that follows:

```
SELECT UOW_ID, SUBSTR(STMT_TEXT, 1,70) AS STMT_TEXT FROM ACTIVITYSTMT_ACTEVMON
WHERE UOW_ID=11
```

In this case, the event monitor is named `actevmon`.

2. Run the query. The preceding query might return results similar to the ones that follow:

```
UOW_ID    STMT_TEXT
-----
11 select * from gosaleshr.employee_expense_detail order by expense_date

1 record(s) selected.
```

## Results

### Example

If you want to access the data in the `DETAILS_XML` column of the `ACTIVITY` table, you can use any of the interfaces that are provided with the DB2 product for this purpose. For example, to see metrics information that an activity event monitor collects for a unit of work, you might use a statement such as the one that follows:

```
SELECT SUBSTR(B.METRIC_NAME, 1, 20) METRIC_NAME, B.VALUE
FROM ACTIVITY_ACTEVMON AS A,
TABLE(MON_FORMAT_XML_METRICS_BY_ROW(A.DETAILS_XML)) AS B
WHERE UOW_ID=23
ORDER BY B.VALUE DESC
```

This statement returns all of the activity metrics collected for the unit of work with the `UOW_ID` of 23:

| METRIC_NAME          | VALUE  |
|----------------------|--------|
| TOTAL_CPU_TIME       | 140625 |
| ROWS_READ            | 977    |
| TOTAL_ACT_TIME       | 880    |
| STMT_EXEC_TIME       | 880    |
| COORD_STMT_EXEC_TIME | 880    |
| TOTAL_SECTION_PROC_T | 880    |
| TOTAL_SECTION_TIME   | 880    |
| :                    |        |
| :                    |        |
| FCM_TQ_SEND_WAITS_TO | 0      |
| FCM_MESSAGE_SEND_WAI | 0      |
| FCM_SEND_WAITS_TOTAL | 0      |
| FCM_RECV_WAITS_TOTAL | 0      |

92 record(s) selected.

For more information about working with XML data returned by event monitors, see “Interfaces that return monitor data in XML documents” on page 13.

### Example: Capturing activity information related to the execution of a statement

If you identify a statement that is taking a long time to execute, you can define a threshold that causes an activity event monitor to capture information about the execution of that statement when the threshold is exceeded.

You can then correlate statement execution information with information collected by the activity event monitor to view activity metrics that can help you understand what might be causing the slowdown.

### Before you begin

Before capturing activity information, you must identify the statement in question; for example, a user or application developer might complain that a specific statement runs longer than expected. Or, you might identify a statement that is taking longer to run by using the package cache event monitor.

### About this task

In this example, the query that is being investigated runs as part of an application. The query is as follows:

```
SELECT DISTINCT PARTS_BIN FROM STOCK WHERE PART_NUMBER = ?
```

One possible reason for a slowdown might be unfavorable data distribution. For example, if the STOCK table has only a few rows for most part numbers, but has several thousand for one particular part number, it takes longer to run this SELECT statement. The example that follows shows how you can retrieve the actual values that are processed for the parameter marker (“?”) by the activity associated with the preceding query.

### Procedure

To test the hypothesis that unfavorable data distribution is the cause of the slow-running query, you can create a threshold for the statement in question. Then you can use the threshold and activity event monitors to capture information about

the execution of that particular statement. From this information, you can determine the actual value that was processed by the query that ran for longer than expected.

1. Create a threshold for the statement in question, specifying that a threshold violation event occurs when the statement runs for longer than 10 seconds:

```
CREATE THRESHOLD TH1
  FOR STATEMENT TEXT 'SELECT DISTINCT PARTS_BIN
  FROM STOCK WHERE PART_NUMBER = ?' ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ACTIVITYTOTALTIME > 10 SECONDS
  COLLECT ACTIVITY DATA WITH DETAILS, SECTION AND VALUES
  CONTINUE
```

2. Create a threshold event monitor to record threshold violations:

```
CREATE EVENT MONITOR STMT_THRESH_VIOLATIONS
  FOR THRESHOLD VIOLATIONS
  WRITE TO TABLE
  AUTOSTART
```

3. Create an activity event monitor to record detailed activity information:

```
CREATE EVENT MONITOR ACTIVITIES
  FOR ACTIVITIES
  WRITE TO TABLE
```

4. Enable the new event monitors:

```
SET EVENT MONITOR ACTIVITIES STATE 1
SET EVENT MONITOR STMT_THRESH_VIOLATIONS STATE 1
```

5. Run the application that executes the statement. If a threshold violation occurs, the threshold violations event monitor STMT\_THRESH\_VIOLATIONS records information about the threshold violation; information about the activity associated with the threshold violation is recorded by the activity event monitor ACTIVITIES.

6. To determine whether a threshold violation occurred, query the number of violations recorded by the threshold event monitor for the threshold TH1 defined in step 1. To perform this query, join the view SYSCAT.THRESHOLDS with the table that was produced by the thresholds event monitor that contains the threshold violation information. This join is necessary because the threshold name TH1 is maintained in SYSCAT.THRESHOLDS:

```
SELECT COUNT(1) NUM_VIOLATIONS
  FROM THRESHOLDVIOLATIONS_DB2THRESHOLDVIOLATIONS T
  JOIN SYSCAT.THRESHOLDS S ON T.THRESHOLDID = S.THRESHOLDID
  WHERE S.THRESHOLDNAME = 'TH1';
```

```
NUM_VIOLATIONS
```

```
-----
```

```
1
```

```
1 record(s) selected.
```

In this case, there was one threshold violation; one execution of the statement identified in step 1 ran for longer than 10 seconds.

7. Examine the data (the part number) that is represented by the parameter marker ("??") in the statement that you identified in 1. In the following example, the SELECT statement retrieves the value for the parameter marker (represented by STMT\_VALUE\_DATA in the SQL that follows) from one of the ACTIVITYVALS tables that the activity event monitor produces:

```
SELECT SUBSTR(V.STMT_VALUE_DATA, 1, 80) PARAM_MARKER_VALUE
  FROM ACTIVITYVALS_ACTIVITIES V
  JOIN THRESHOLDVIOLATIONS_STMT_THRESH_VIOLATIONS T
  ON T.APPL_ID = V.APPL_ID
  AND T.UOW_ID = V.UOW_ID
```



## System metrics

*System metrics* are request monitor elements that capture measurement information about the system. They include the time-spent monitor elements, as well as monitor elements that are counters, such as the monitor element deadlocks. Collection of these metrics can be configured for the entire database, or for specific service classes. The values in the system metrics monitor elements generally start at 0, and continue to accumulate until the next database activation. System metrics are collected as part of the details\_xml monitor element in the event\_scstats and event\_wlstats logical data groups.

## Workload management statistics

Statistics are maintained for workload manager objects including service classes, work classes, workloads, and threshold queues. These statistics reside in memory and can be viewed in real-time using workload manager statistics table functions, or the statistics can be collected and sent to a statistics event monitor where they can be viewed later for historical analysis. By default, a minimal set of statistics are collected for each workload manager object. You can modify the scope of statistics collection using the clauses in the CREATE or ALTER statements for the various workload manager objects. Workload management statistics are reset to 0 after each collection interval.

The metrics collected by the statistics event monitor are the same set of metrics that are reported by the MON\_GET\_SERVICE\_SUBCLASS\_DETAILS and MON\_GET\_WORKLOAD\_DETAILS table functions. Note, however, that, the XML document in the DETAILS column of the tables returned by these two table functions contain many other monitor elements in addition to the system metrics monitor elements.

The tables produced by a write-to-table statistics event monitor are described in “Information written to tables for a statistics event monitor” on page 274. The system metrics are collected in the XML document details\_xml in the DETAILS\_XML columns of the SCSTATS and WLSTATS tables. These XML documents include dozens of metrics that reflect a variety of different system measurements. The schema for the XML document that is returned in the DETAILS\_XML column is available in the file sql11lib/misc/DB2MonCommon.xsd. The top level element is system\_metrics.

**Remember:** The data returned for metrics monitor elements in the details\_xml document produced by the statistics event monitor accumulate **continuously** from the time that the database is activated. By contrast, the workload management statistics monitor elements that appear in the other columns of the tables produced by the statistics event monitor **are reset to 0** at each collection interval.

The collection of system metrics monitor elements by the statistics event monitor is controlled through two mechanisms:

- To collect metrics for specific service classes, you can collect system metrics by specifying the COLLECT REQUEST METRICS clause of CREATE or ALTER SERVICE CLASS statements.
- To collect metrics for the entire database, use the **mon\_req\_metrics** database configuration parameter. For example, to collect BASE-level metrics, use the following command:

```
db2 update db cfg using mon_req_metrics base
```

System metrics monitor elements are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request monitor element collection enabled, or if system metrics collection is enabled for the entire database. If system metrics collection is disabled at the database level, as well as for a service superclass, the metrics reported in the DETAILS\_XML document stop increasing (or remain at 0 if request metrics were disabled at database activation time).

### Data generated by statistics event monitors

You can choose to have the output from a statistics event monitor written to regular tables, files or named pipes.

Regardless of the output format you choose, all data captured by statistics event monitors comes from one of five logical data groups:

- event\_scstats
- event\_wcstats
- event\_wlstats
- event\_qstats
- event\_histogrambin

If you choose to have the statistics event monitor data written to regular tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

#### Information written to tables for a statistics event monitor:

Information written by a statistics event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for a statistics event monitor, by default, five tables are produced, each containing monitor elements from one or more logical data groups:

*Table 53. Tables produced by STATISTICS write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name              | Logical data groups reported  |
|---------------------------------|---|
| QSTATS_ <i>evmon-name</i>       | “event_qstats logical data group” on page 61  |
| SCSTATS_ <i>evmon-name</i>      | “event_scstats logical data group” on page 62   |
| HISTOGRAMBIN_ <i>evmon-name</i> | “event_histogrambin logical data group” on page 61  |
| WCSTATS_ <i>evmon-name</i>      | “event_wcstats logical data group” on page 68   |
| WLSTATS_ <i>evmon-name</i>      | “event_wlstats logical data group” on page 69   |
| CONTROL_ <i>evmon-name</i>      | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

## Tables produced

Table 54. Information returned for a statistics event monitor: Default table name: QSTATS\_evmon-name

| Column Name             | Data Type    | Description   |
|-------------------------|--------------|---|
| PARTITION_KEY           | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| LAST_WLM_RESET          | TIMESTAMP    | last_wlm_reset - Time of last reset                             |
| MON_INTERVAL_ID         | BIGINT       | mon_interval_id - Monitor interval identifier                   |
| PARTITION_NUMBER        | SMALLINT     | partition_number - Partition number                             |
| QUEUE_ASSIGNMENTS_TOTAL | BIGINT       | queue_assignments_total - Queue assignments total               |
| QUEUE_SIZE_TOP          | INTEGER      | queue_size_top - Queue size top                                 |
| QUEUE_TIME_TOTAL        | BIGINT       | queue_time_total - Queue time total                             |
| SERVICE_SUBCLASS_NAME   | VARCHAR(128) | service_subclass_name - Service subclass name                   |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name               |
| STATISTICS_TIMESTAMP    | TIMESTAMP    | statistics_timestamp - Statistics timestamp                     |
| THRESHOLD_DOMAIN        | VARCHAR(64)  | threshold_domain - Threshold domain                             |
| THRESHOLD_NAME          | VARCHAR(128) | threshold_name - Threshold name                                 |
| THRESHOLD_PREDICATE     | VARCHAR(64)  | threshold_predicate - Threshold predicate                       |
| THRESHOLDID             | INTEGER      | thresholdid - Threshold ID                                      |
| WORK_ACTION_SET_NAME    | VARCHAR(128) | work_action_set_name - Work action set name                     |
| WORK_CLASS_NAME         | VARCHAR(128) | work_class_name - Work class name                               |

Table 55. Information returned for a statistics event monitor: Table name: SCSTATS\_evmon-name

| Column Name               | Data Type | Description  |
|---------------------------|-----------|--|
| PARTITION_KEY             | INTEGER   | "partition_key - Partitioning key monitor element" on page 1016            |
| ACT_CPU_TIME_TOP          | BIGINT    | act_cpu_time_top - Activity CPU time top                                   |
| ACT_REMAPPED_IN           | BIGINT    | act_remapped_in - Activities remapped in                                   |
| ACT_REMAPPED_OUT          | BIGINT    | act_remapped_out - Activities remapped out                                 |
| ACT_ROWS_READ_TOP         | BIGINT    | act_rows_read_top - Activity rows read top                                 |
| ACT_THROUGHPUT            | BIGINT    | act_throughput - Activity throughput                                       |
| AGG_TEMP_TABLESPACE_TOP   | BIGINT    | agg_temp_tablespace_top - Aggregate temporary table space top              |
| APP_ACT_ABORTED_TOTAL     | BIGINT    | app_act_aborted_total - Total failed external coordinator activities       |
| APP_ACT_COMPLETED_TOTAL   | BIGINT    | app_act_completed_total - Total successful external coordinator activities |
| APP_ACT_REJECTED_TOTAL    | BIGINT    | app_act_rejected_total - Total rejected external coordinator activities    |
| CONCURRENT_ACT_TOP        | INTEGER   | concurrent_act_top - Concurrent activity top                               |
| CONCURRENT_CONNECTION_TOP | INTEGER   | concurrent_connection_top - Concurrent connection top                      |
| CONCURRENT_WLO_TOP        | INTEGER   | concurrent_wlo_top - Concurrent workload occurrences top                   |
| COORD_ACT_ABORTED_TOTAL   | BIGINT    | coord_act_aborted_total - Coordinator activities aborted total             |

Table 55. Information returned for a statistics event monitor: Table name: SCSTATS\_evmon-name (continued)

| Column Name                     | Data Type    | Description  |
|---------------------------------|--------------|--|
| COORD_ACT_COMPLETED_TOTAL       | BIGINT       | coord_act_completed_total - Coordinator activities completed total   |
| COORD_ACT_EST_COST_AVG          | BIGINT       | coord_act_est_cost_avg - Coordinator activity estimated cost average   |
| COORD_ACT_EXEC_TIME_AVG         | BIGINT       | coord_act_exec_time_avg - Coordinator activities execution time average  |
| COORD_ACT_INTERARRIVAL_TIME_AVG | BIGINT       | coord_act_interarrival_time_avg - Coordinator activity arrival time average  |
| COORD_ACT_LIFETIME_AVG          | BIGINT       | coord_act_lifetime_avg - Coordinator activity lifetime average   |
| COORD_ACT_LIFETIME_TOP          | BIGINT       | coord_act_lifetime_top - Coordinator activity lifetime top   |
| COORD_ACT_QUEUE_TIME_AVG        | BIGINT       | coord_act_queue_time_avg - Coordinator activity queue time average   |
| COORD_ACT_REJECTED_TOTAL        | BIGINT       | coord_act_rejected_total - Coordinator activities rejected total   |
| COST_ESTIMATE_TOP               | BIGINT       | cost_estimate_top - Cost estimate top  |
| CPU_UTILIZATION                 | BIGINT       | cpu_utilization - CPU utilization  |
| DETAILS_XML                     | BLOB(0)      | <b>Note:</b> Metrics reported in this document for the default subclass SYSDEFAULTSUBCLASS under the superclass SYSDEFAULTSYSTEMCLASS have a value of 0. |
| LAST_WLM_RESET                  | TIMESTAMP    | last_wlm_reset - Time of last reset  |
| MON_INTERVAL_ID                 | BIGINT       | mon_interval_id - Monitor interval identifier  |
| PARTITION_NUMBER                | SMALLINT     | partition_number - Partition number  |
| REQUEST_EXEC_TIME_AVG           | BIGINT       | request_exec_time_avg - Request execution time average   |
| ROWS_RETURNED_TOP               | BIGINT       | rows_returned_top - Actual rows returned top   |
| SERVICE_CLASS_ID                | INTEGER      | service_class_id - Service class ID  |
| SERVICE_SUBCLASS_NAME           | VARCHAR(128) | service_subclass_name - Service subclass name  |
| SERVICE_SUPERCLASS_NAME         | VARCHAR(128) | service_superclass_name - Service superclass name  |
| STATISTICS_TIMESTAMP            | TIMESTAMP    | statistics_timestamp - Statistics timestamp  |
| TEMP_TABLESPACE_TOP             | BIGINT       | temp_tablespace_top - Temporary table space top  |
| TOTAL_CPU_TIME                  | BIGINT       | total_cpu_time - Total CPU time  |
| TOTAL_DISP_RUN_QUEUE_TIME       | BIGINT       | total_disp_run_queue_time - Total dispatcher run queue time  |
| UOW_COMPLETED_TOTAL             | BIGINT       | uow_completed_total - Total completed units of work  |
| UOW_LIFETIME_AVG                | BIGINT       | uow_lifetime_avg - Unit of work lifetime average   |
| UOW_THROUGHPUT                  | BIGINT       | uow_throughput - Unit of work throughput   |
| UOW_TOTAL_TIME_TOP              | BIGINT       | uow_total_time_top - UOW total time top  |

Table 56. Information returned for a statistics event monitor: Table name: HISTOGRAMBIN\_evmon-name

| Column Name          | Data Type   | Description   |
|----------------------|-------------|---|
| PARTITION_KEY        | INTEGER     | "partition_key - Partitioning key monitor element" on page 1016 |
| BIN_ID               | INTEGER     | bin_id - Histogram bin identifier                               |
| BOTTOM               | BIGINT      | bottom - Histogram bin bottom                                   |
| HISTOGRAM_TYPE       | VARCHAR(64) | histogram_type - Histogram type                                 |
| MON_INTERVAL_ID      | BIGINT      | mon_interval_id - Monitor interval identifier                   |
| NUMBER_IN_BIN        | BIGINT      | number_in_bin - Number in bin                                   |
| PARTITION_NUMBER     | SMALLINT    | partition_number - Partition number                             |
| SERVICE_CLASS_ID     | INTEGER     | service_class_id - Service class ID                             |
| STATISTICS_TIMESTAMP | TIMESTAMP   | statistics_timestamp - Statistics timestamp                     |
| TOP                  | BIGINT      | top - Histogram bin top   |
| WORK_ACTION_SET_ID   | INTEGER     | work_action_set_id - Work action set ID                         |
| WORK_CLASS_ID        | INTEGER     | work_class_id - Work class ID                                   |
| WORKLOAD_ID          | INTEGER     | workload_id - Workload ID                                       |

Table 57. Information returned for a statistics event monitor: Table name: WCSTATS\_evmon-name

| Column Name                     | Data Type | Description   |
|---------------------------------|-----------|---|
| PARTITION_KEY                   | INTEGER   | "partition_key - Partitioning key monitor element" on page 1016         |
| ACT_CPU_TIME_TOP                | BIGINT    | act_cpu_time_top - Activity CPU time top                                |
| ACT_ROWS_READ_TOP               | BIGINT    | act_rows_read_top - Activity rows read top                              |
| ACT_TOTAL                       | BIGINT    | act_total - Activities total  |
| COORD_ACT_EST_COST_AVG          | BIGINT    | coord_act_est_cost_avg - Coordinator activity estimated cost average    |
| COORD_ACT_EXEC_TIME_AVG         | BIGINT    | coord_act_exec_time_avg - Coordinator activities execution time average |
| COORD_ACT_INTERARRIVAL_TIME_AVG | BIGINT    | coord_act_exec_time_avg - Coordinator activities execution time average |
| COORD_ACT_LIFETIME_AVG          | BIGINT    | coord_act_lifetime_avg - Coordinator activity lifetime average          |
| COORD_ACT_LIFETIME_TOP          | BIGINT    | coord_act_lifetime_top - Coordinator activity lifetime top              |
| COORD_ACT_QUEUE_TIME_AVG        | BIGINT    | coord_act_queue_time_avg - Coordinator activity queue time average      |
| COST_ESTIMATE_TOP               | BIGINT    | cost_estimate_top - Cost estimate top                                   |
| LAST_WLM_RESET                  | TIMESTAMP | last_wlm_reset - Time of last reset                                     |
| MON_INTERVAL_ID                 | BIGINT    | mon_interval_id - Monitor interval identifier                           |
| PARTITION_NUMBER                | SMALLINT  | partition_number - Partition number                                     |
| ROWS_RETURNED_TOP               | BIGINT    | rows_returned_top - Actual rows returned top                            |
| STATISTICS_TIMESTAMP            | TIMESTAMP | statistics_timestamp - Statistics timestamp                             |
| TEMP_TABLESPACE_TOP             | BIGINT    | temp_tablespace_top - Temporary table space top                         |
| WORK_ACTION_SET_ID              | INTEGER   | work_action_set_id - Work action set ID                                 |

Table 57. Information returned for a statistics event monitor: Table name: WCSTATS\_evmon-name (continued)

| Column Name          | Data Type    | Description                                 |
|----------------------|--------------|---|
| WORK_ACTION_SET_NAME | VARCHAR(128) | work_action_set_name - Work action set name |
| WORK_CLASS_ID        | INTEGER      | work_class_id - Work class ID               |
| WORK_CLASS_NAME      | VARCHAR(128) | work_class_name - Work class name           |

Table 58. Information returned for a statistics event monitor: Table name: WLSTATS\_evmon-name

| Column Name                     | Data Type | Description   |
|---------------------------------|-----------|---|
| PARTITION_KEY                   | INTEGER   | "partition_key - Partitioning key monitor element" on page 1016   |
| ACT_CPU_TIME_TOP                | BIGINT    | act_cpu_time_top - Activity CPU time top  |
| ACT_ROWS_READ_TOP               | BIGINT    | act_rows_read_top - Activity rows read top  |
| ACT_THROUGHPUT                  | BIGINT    | act_throughput - Activity throughput  |
| APP_ACT_ABORTED_TOTAL           | BIGINT    | app_act_aborted_total - Total failed external coordinator activities                                      |
| APP_ACT_COMPLETED_TOTAL         | BIGINT    | app_act_completed_total - Total successful external coordinator activities                                |
| APP_ACT_REJECTED_TOTAL          | BIGINT    | app_act_rejected_total - Total rejected external coordinator activities                                   |
| CONCURRENT_WLO_ACT_TOP          | INTEGER   | concurrent_wlo_act_top - Concurrent WLO activity top  |
| CONCURRENT_WLO_TOP              | INTEGER   | concurrent_wlo_top - Concurrent workload occurrences top  |
| COORD_ACT_ABORTED_TOTAL         | BIGINT    | coord_act_aborted_total - Coordinator activities aborted total  |
| COORD_ACT_COMPLETED_TOTAL       | BIGINT    | coord_act_completed_total - Coordinator activities completed total  |
| COORD_ACT_EST_COST_AVG          | BIGINT    | coord_act_est_cost_avg - Coordinator activity estimated cost average                                      |
| COORD_ACT_EXEC_TIME_AVG         | BIGINT    | coord_act_exec_time_avg - Coordinator activities execution time average                                   |
| COORD_ACT_INTERARRIVAL_TIME_AVG | BIGINT    | "coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element" on page 744 |
| COORD_ACT_LIFETIME_AVG          | BIGINT    | coord_act_lifetime_avg - Coordinator activity lifetime average  |
| COORD_ACT_LIFETIME_TOP          | BIGINT    | coord_act_lifetime_top - Coordinator activity lifetime top  |
| COORD_ACT_QUEUE_TIME_AVG        | BIGINT    | coord_act_queue_time_avg - Coordinator activity queue time average  |
| COORD_ACT_REJECTED_TOTAL        | BIGINT    | coord_act_rejected_total - Coordinator activities rejected total  |
| COST_ESTIMATE_TOP               | BIGINT    | cost_estimate_top - Cost estimate top   |
| CPU_UTILIZATION                 | BIGINT    | cpu_utilization - CPU utilization   |
| DETAILS_XML                     | BLOB(0)   |   |
| LAST_WLM_RESET                  | TIMESTAMP | last_wlm_reset - Time of last reset   |

Table 58. Information returned for a statistics event monitor: Table name: WLSTATS\_evmon-name (continued)

| Column Name               | Data Type    | Description   |
|---------------------------|--------------|---|
| LOCK_WAIT_TIME_GLOBAL_TOP | BIGINT       | lock_wait_time_global_top - Top global lock wait time       |
| LOCK_WAIT_TIME_TOP        | BIGINT       | lock_wait_time_top - Lock wait time top                     |
| MON_INTERVAL_ID           | BIGINT       | mon_interval_id - Monitor interval identifier               |
| PARTITION_NUMBER          | SMALLINT     | partition_number - Partition number                         |
| ROWS_RETURNED_TOP         | BIGINT       | rows_returned_top - Actual rows returned top                |
| STATISTICS_TIMESTAMP      | TIMESTAMP    | statistics_timestamp - Statistics timestamp                 |
| TEMP_TABLESPACE_TOP       | BIGINT       | temp_tablespace_top - Temporary table space top             |
| TOTAL_CPU_TIME            | BIGINT       | total_cpu_time - Total CPU time                             |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT       | total_disp_run_queue_time - Total dispatcher run queue time |
| UOW_COMPLETED_TOTAL       | BIGINT       | uow_completed_total - Total completed units of work         |
| UOW_LIFETIME_AVG          | BIGINT       | uow_lifetime_avg - Unit of work lifetime average            |
| UOW_THROUGHPUT            | BIGINT       | uow_throughput - Unit of work throughput                    |
| UOW_TOTAL_TIME_TOP        | BIGINT       | uow_total_time_top - UOW total time top                     |
| WLO_COMPLETED_TOTAL       | BIGINT       | wlo_completed_total - Workload occurrences completed total  |
| WORKLOAD_ID               | INTEGER      | workload_id - Workload ID                                   |
| WORKLOAD_NAME             | VARCHAR(128) | workload_name - Workload name                               |

Table 59. Information returned for a statistics event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | "partition_key - Partitioning key monitor element" on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

**Information written to XML for system\_metrics and activity\_metrics monitor elements:**

The **activity\_metrics** monitor element is reported in the MON\_GET\_ACTIVITY\_DETAILS table function, the MON\_GET\_PKG\_CACHE\_STMT\_DETAILS table function, and the activity event monitor. The **system\_metrics** monitor element is reported in the MON\_GET\_CONNECTION\_DETAILS, MON\_GET\_UNIT\_OF\_WORK\_DETAILS, MON\_GET\_SERVICE\_SUBCLASS\_DETAILS, MON\_GET\_WORKLOAD\_DETAILS table functions, and the statistics event monitor. This is also documented in the sql/lib/misc/DB2MonCommon.xsd file.

## system\_metrics

System level metrics.

**Element content:** ( "wlm\_queue\_time\_total" on page 300, "wlm\_queue\_assignments\_total" on page 301, "fcm\_tq\_recv\_wait\_time" on page 301, "fcm\_message\_recv\_wait\_time" on page 301, "fcm\_tq\_send\_wait\_time" on page 301, "fcm\_message\_send\_wait\_time" on page 301, "agent\_wait\_time" on page 285, "agent\_waits\_total" on page 285, "lock\_wait\_time" on page 302, "lock\_waits" on page 302, "direct\_read\_time" on page 302, "direct\_read\_reqs" on page 302, "direct\_write\_time" on page 303, "direct\_write\_reqs" on page 303, "log\_buffer\_wait\_time" on page 303, "num\_log\_buffer\_full" on page 303, "log\_disk\_wait\_time" on page 304, "log\_disk\_waits\_total" on page 304, "tcpip\_recv\_wait\_time" on page 285, "tcpip\_recvs\_total" on page 285, "client\_idle\_wait\_time" on page 285, "ipc\_recv\_wait\_time" on page 286, "ipc\_recvs\_total" on page 286, "ipc\_send\_wait\_time" on page 286, "ipc\_sends\_total" on page 286, "tcpip\_send\_wait\_time" on page 287, "tcpip\_sends\_total" on page 287, "pool\_write\_time" on page 304, "pool\_read\_time" on page 304, "audit\_file\_write\_wait\_time" on page 304, "audit\_file\_writes\_total" on page 305, "audit\_subsystem\_wait\_time" on page 305, "audit\_subsystem\_waits\_total" on page 305, "diaglog\_write\_wait\_time" on page 305, "diaglog\_writes\_total" on page 306, "fcm\_send\_wait\_time" on page 306, "fcm\_recv\_wait\_time" on page 306, "total\_wait\_time" on page 287, "total\_rqst\_time" on page 287, "rqsts\_completed\_total" on page 288, "total\_app\_rqst\_time" on page 288, "app\_rqsts\_completed\_total" on page 288, "total\_section\_sort\_proc\_time" on page 307, "total\_section\_sort\_time" on page 307, "total\_section\_sorts" on page 307, "rows\_read" on page 307, "rows\_modified" on page 308, "pool\_data\_l\_reads" on page 308, "pool\_index\_l\_reads" on page 308, "pool\_temp\_data\_l\_reads" on page 308, "pool\_temp\_index\_l\_reads" on page 309, "pool\_xda\_l\_reads" on page 309, "pool\_temp\_xda\_l\_reads" on page 309, "total\_cpu\_time" on page 309, "act\_completed\_total" on page 288, "pool\_data\_p\_reads" on page 310, "pool\_temp\_data\_p\_reads" on page 310, "pool\_xda\_p\_reads" on page 310, "pool\_temp\_xda\_p\_reads" on page 310, "pool\_index\_p\_reads" on page 310, "pool\_temp\_index\_p\_reads" on page 311, "pool\_data\_writes" on page 311, "pool\_xda\_writes" on page 311, "pool\_index\_writes" on page 311, "direct\_reads" on page 312, "direct\_writes" on page 312, "rows\_returned" on page 312, "deadlocks" on page 312, "lock\_timeouts" on page 313, "lock\_escals" on page 313, "fcm\_sends\_total" on page 313, "fcm\_recvs\_total" on page 313, "fcm\_send\_volume" on page 313, "fcm\_recv\_volume" on page 314, "fcm\_message\_sends\_total" on page 314, "fcm\_message\_recvs\_total" on page 314, "fcm\_message\_send\_volume" on page 314, "fcm\_message\_recv\_volume" on page 315, "fcm\_tq\_sends\_total" on page 315, "fcm\_tq\_recvs\_total" on page 315, "fcm\_tq\_send\_volume" on page 315, "fcm\_tq\_recv\_volume" on page 316, "tq\_tot\_send\_spills" on page 316, "tcpip\_send\_volume" on page 288, "tcpip\_recv\_volume" on page 289, "ipc\_send\_volume" on page 289, "ipc\_recv\_volume" on page 289, "post\_threshold\_sorts" on page 316, "post\_shrthreshold\_sorts" on page 316, "sort\_overflows" on page 316, "audit\_events\_total" on page 317, "total\_rqst\_mapped\_in" on page 289 {zero or one times (?)}, "total\_rqst\_mapped\_out" on page 290 {zero or one times (?)}, "act\_rejected\_total" on page 290, "act\_aborted\_total" on page 290, "total\_sorts" on page 317, "total\_routine\_time" on page 319, "total\_compile\_proc\_time" on page 290, "total\_compile\_time" on page 291, "total\_compilations" on page 291, "total\_implicit\_compile\_proc\_time" on page 291, "total\_implicit\_compile\_time" on page 291, "total\_implicit\_compilations" on page 291, "total\_runstats\_proc\_time" on page 292, "total\_runstats\_time" on page 292, "total\_runstats" on page 292,

"total\_reorg\_proc\_time" on page 292, "total\_reorg\_time" on page 293,  
 "total\_reorgs" on page 293, "total\_load\_proc\_time" on page 293, "total\_load\_time"  
 on page 293, "total\_loads" on page 294, "total\_section\_proc\_time" on page 318,  
 "total\_section\_time" on page 318, "total\_app\_section\_executions" on page 319,  
 "total\_commit\_proc\_time" on page 294, "total\_commit\_time" on page 294,  
 "total\_app\_commits" on page 294, "total\_rollback\_proc\_time" on page 294,  
 "total\_rollback\_time" on page 295, "total\_app\_rollbacks" on page 295,  
 "total\_routine\_user\_code\_proc\_time" on page 319, "total\_routine\_user\_code\_time"  
 on page 319, "thresh\_violations" on page 319, "num\_lw\_thresh\_exceeded" on page  
 320, "total\_routine\_invocations" on page 320, "int\_commits" on page 295,  
 "int\_rollbacks" on page 295, "cat\_cache\_inserts" on page 296, "cat\_cache\_lookups"  
 on page 296, "pkg\_cache\_inserts" on page 296, "pkg\_cache\_lookups" on page 296,  
 "act\_rqsts\_total" on page 297, "total\_act\_wait\_time" on page 306, "total\_act\_time"  
 on page 307, "lock\_wait\_time\_global" on page 320, "lock\_waits\_global" on page  
 320, "reclaim\_wait\_time" on page 321, "spacemappage\_reclaim\_wait\_time" on  
 page 321, "lock\_timeouts\_global" on page 321, "lock\_escals\_maxlocks" on page  
 321, "lock\_escals\_locklist" on page 322, "lock\_escals\_global" on page 322,  
 "cf\_wait\_time" on page 322, "cf\_waits" on page 322, "pool\_data\_gbp\_l\_reads" on  
 page 322, "pool\_data\_gbp\_p\_reads" on page 323, "pool\_data\_lbp\_pages\_found" on  
 page 323, "pool\_data\_gbp\_invalid\_pages" on page 323, "pool\_index\_gbp\_l\_reads"  
 on page 323, "pool\_index\_gbp\_p\_reads" on page 324,  
 "pool\_index\_lbp\_pages\_found" on page 324, "pool\_index\_gbp\_invalid\_pages" on  
 page 324, "pool\_xda\_gbp\_l\_reads" on page 324, "pool\_xda\_gbp\_p\_reads" on page  
 325, "pool\_xda\_lbp\_pages\_found" on page 325, "pool\_xda\_gbp\_invalid\_pages" on  
 page 325, "evmon\_wait\_time" on page 325, "evmon\_waits\_total" on page 325,  
 "total\_extended\_latch\_wait\_time" on page 326, "total\_extended\_latch\_waits" on  
 page 326, "total\_stats\_fabrication\_proc\_time" on page 297,  
 "total\_stats\_fabrication\_time" on page 297, "total\_stats\_fabrications" on page 297,  
 "total\_sync\_runstats\_proc\_time" on page 297, "total\_sync\_runstats\_time" on page  
 298, "total\_sync\_runstats" on page 298, "total\_disp\_run\_queue\_time" on page 326,  
 "pool\_queued\_async\_data\_reqs" on page 326, "pool\_queued\_async\_index\_reqs" on  
 page 327, "pool\_queued\_async\_xda\_reqs" on page 327,  
 "pool\_queued\_async\_temp\_data\_reqs" on page 327,  
 "pool\_queued\_async\_temp\_index\_reqs" on page 327,  
 "pool\_queued\_async\_temp\_xda\_reqs" on page 328,  
 "pool\_queued\_async\_other\_reqs" on page 328, "pool\_queued\_async\_data\_pages"  
 on page 328, "pool\_queued\_async\_index\_pages" on page 328,  
 "pool\_queued\_async\_xda\_pages" on page 329,  
 "pool\_queued\_async\_temp\_data\_pages" on page 329,  
 "pool\_queued\_async\_temp\_index\_pages" on page 329,  
 "pool\_queued\_async\_temp\_xda\_pages" on page 329,  
 "pool\_failed\_async\_data\_reqs" on page 330, "pool\_failed\_async\_index\_reqs" on  
 page 330, "pool\_failed\_async\_xda\_reqs" on page 330,  
 "pool\_failed\_async\_temp\_data\_reqs" on page 330,  
 "pool\_failed\_async\_temp\_index\_reqs" on page 331,  
 "pool\_failed\_async\_temp\_xda\_reqs" on page 331, "pool\_failed\_async\_other\_reqs"  
 on page 331, "app\_act\_completed\_total" on page 298, "app\_act\_aborted\_total" on  
 page 298, "app\_act\_rejected\_total" on page 299, "total\_peds" on page 331,  
 "disabled\_peds" on page 332, "post\_threshold\_peds" on page 332, "total\_peas" on  
 page 332, "post\_threshold\_peas" on page 332, "tq\_sort\_heap\_requests" on page  
 332, "tq\_sort\_heap\_rejections" on page 333, "total\_connect\_request\_proc\_time" on  
 page 299, "total\_connect\_request\_time" on page 299, "total\_connect\_requests" on  
 page 299, "total\_connect\_authentication\_proc\_time" on page 300,  
 "total\_connect\_authentication\_time" on page 300, "total\_connect\_authentications"  
 on page 300, "prefetch\_wait\_time" on page 333, "prefetch\_waits" on page 333,  
 "pool\_data\_gbp\_indep\_pages\_found\_in\_lbp" on page 333,

“pool\_index\_gbp\_indep\_pages\_found\_in\_lbp” on page 334,  
 “pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp” on page 334, ANY content ( skip )  
 {zero or more (\*) )

**Attributes:**

| QName                            | Type    | Fixed | Default | Use      | Annotation |
|----------------------------------|---------|-------|---------|----------|------------|
| release                          | xs:long |       |         | required |            |
| ANY attribute from ANY namespace |         |       |         |          |            |

**activity\_metrics**

Activity level metrics.

**Element content:** ( “wlm\_queue\_time\_total” on page 300,  
 “wlm\_queue\_assignments\_total” on page 301, “fcm\_tq\_rcv\_wait\_time” on page 301,  
 “fcm\_message\_rcv\_wait\_time” on page 301, “fcm\_tq\_send\_wait\_time” on page 301,  
 “fcm\_message\_send\_wait\_time” on page 301, “lock\_wait\_time” on page 302,  
 “lock\_waits” on page 302, “direct\_read\_time” on page 302, “direct\_read\_reqs”  
 on page 302, “direct\_write\_time” on page 303, “direct\_write\_reqs” on page 303,  
 “log\_buffer\_wait\_time” on page 303, “num\_log\_buffer\_full” on page 303,  
 “log\_disk\_wait\_time” on page 304, “log\_disk\_waits\_total” on page 304,  
 “pool\_write\_time” on page 304, “pool\_read\_time” on page 304,  
 “audit\_file\_write\_wait\_time” on page 304, “audit\_file\_writes\_total” on page 305,  
 “audit\_subsystem\_wait\_time” on page 305, “audit\_subsystem\_waits\_total” on page 305,  
 “diaglog\_write\_wait\_time” on page 305, “diaglog\_writes\_total” on page 306,  
 “fcm\_send\_wait\_time” on page 306, “fcm\_rcv\_wait\_time” on page 306,  
 “total\_act\_wait\_time” on page 306, “total\_section\_sort\_proc\_time” on page 307,  
 “total\_section\_sort\_time” on page 307, “total\_section\_sorts” on page 307,  
 “total\_act\_time” on page 307, “rows\_read” on page 307, “rows\_modified” on page 308,  
 “pool\_data\_l\_reads” on page 308, “pool\_index\_l\_reads” on page 308,  
 “pool\_temp\_data\_l\_reads” on page 308, “pool\_temp\_index\_l\_reads” on page 309,  
 “pool\_xda\_l\_reads” on page 309, “pool\_temp\_xda\_l\_reads” on page 309,  
 “total\_cpu\_time” on page 309, “pool\_data\_p\_reads” on page 310,  
 “pool\_temp\_data\_p\_reads” on page 310, “pool\_xda\_p\_reads” on page 310,  
 “pool\_temp\_xda\_p\_reads” on page 310, “pool\_index\_p\_reads” on page 310,  
 “pool\_temp\_index\_p\_reads” on page 311, “pool\_data\_writes” on page 311,  
 “pool\_xda\_writes” on page 311, “pool\_index\_writes” on page 311, “direct\_reads”  
 on page 312, “direct\_writes” on page 312, “rows\_returned” on page 312,  
 “deadlocks” on page 312, “lock\_timeouts” on page 313, “lock\_escals” on page 313,  
 “fcm\_sends\_total” on page 313, “fcm\_rcvs\_total” on page 313,  
 “fcm\_send\_volume” on page 313, “fcm\_rcv\_volume” on page 314,  
 “fcm\_message\_sends\_total” on page 314, “fcm\_message\_rcvs\_total” on page 314,  
 “fcm\_message\_send\_volume” on page 314, “fcm\_message\_rcv\_volume” on page 315,  
 “fcm\_tq\_sends\_total” on page 315, “fcm\_tq\_rcvs\_total” on page 315,  
 “fcm\_tq\_send\_volume” on page 315, “fcm\_tq\_rcv\_volume” on page 316,  
 “tq\_tot\_send\_spills” on page 316, “post\_threshold\_sorts” on page 316,  
 “post\_shrthreshold\_sorts” on page 316, “sort\_overflows” on page 316,  
 “audit\_events\_total” on page 317, “total\_sorts” on page 317, “stmt\_exec\_time” on  
 page 317, “coord\_stmt\_exec\_time” on page 317 {zero or one times (?)} ,  
 “total\_routine\_non\_sect\_proc\_time” on page 318, “total\_routine\_non\_sect\_time” on  
 page 318, “total\_section\_proc\_time” on page 318, “total\_section\_time” on page 318,  
 “total\_app\_section\_executions” on page 319, “total\_routine\_user\_code\_proc\_time”  
 on page 319

on page 317, "total\_routine\_user\_code\_time" on page 319, "total\_routine\_time" on page 319, "thresh\_violations" on page 319, "num\_lw\_thresh\_exceeded" on page 320, "total\_routine\_invocations" on page 320, "lock\_wait\_time\_global" on page 320, "lock\_waits\_global" on page 320, "reclaim\_wait\_time" on page 321, "spacemappage\_reclaim\_wait\_time" on page 321, "lock\_timeouts\_global" on page 321, "lock\_escals\_maxlocks" on page 321, "lock\_escals\_locklist" on page 322, "lock\_escals\_global" on page 322, "cf\_wait\_time" on page 322, "cf\_waits" on page 322, "pool\_data\_gbp\_l\_reads" on page 322, "pool\_data\_gbp\_p\_reads" on page 323, "pool\_data\_lbp\_pages\_found" on page 323, "pool\_data\_gbp\_invalid\_pages" on page 323, "pool\_index\_gbp\_l\_reads" on page 323, "pool\_index\_gbp\_p\_reads" on page 324, "pool\_index\_lbp\_pages\_found" on page 324, "pool\_index\_gbp\_invalid\_pages" on page 324, "pool\_xda\_gbp\_l\_reads" on page 324, "pool\_xda\_gbp\_p\_reads" on page 325, "pool\_xda\_lbp\_pages\_found" on page 325, "pool\_xda\_gbp\_invalid\_pages" on page 325, "evmon\_wait\_time" on page 325, "evmon\_waits\_total" on page 325, "total\_extended\_latch\_wait\_time" on page 326, "total\_extended\_latch\_waits" on page 326, "total\_disp\_run\_queue\_time" on page 326, "pool\_queued\_async\_data\_reqs" on page 326, "pool\_queued\_async\_index\_reqs" on page 327, "pool\_queued\_async\_xda\_reqs" on page 327, "pool\_queued\_async\_temp\_data\_reqs" on page 327, "pool\_queued\_async\_temp\_index\_reqs" on page 327, "pool\_queued\_async\_temp\_xda\_reqs" on page 328, "pool\_queued\_async\_other\_reqs" on page 328, "pool\_queued\_async\_data\_pages" on page 328, "pool\_queued\_async\_index\_pages" on page 328, "pool\_queued\_async\_xda\_pages" on page 329, "pool\_queued\_async\_temp\_data\_pages" on page 329, "pool\_queued\_async\_temp\_index\_pages" on page 329, "pool\_queued\_async\_temp\_xda\_pages" on page 329, "pool\_failed\_async\_data\_reqs" on page 330, "pool\_failed\_async\_index\_reqs" on page 330, "pool\_failed\_async\_xda\_reqs" on page 330, "pool\_failed\_async\_temp\_data\_reqs" on page 330, "pool\_failed\_async\_temp\_index\_reqs" on page 331, "pool\_failed\_async\_temp\_xda\_reqs" on page 331, "pool\_failed\_async\_other\_reqs" on page 331, "total\_peds" on page 331, "disabled\_peds" on page 332, "post\_threshold\_peds" on page 332, "total\_peas" on page 332, "post\_threshold\_peas" on page 332, "tq\_sort\_heap\_requests" on page 332, "tq\_sort\_heap\_rejections" on page 333, "prefetch\_wait\_time" on page 333, "prefetch\_waits" on page 333, "pool\_data\_gbp\_indep\_pages\_found\_in\_lbp" on page 333, "pool\_index\_gbp\_indep\_pages\_found\_in\_lbp" on page 334, "pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp" on page 334, ANY content ( skip ) {zero or more (\*) )

**Attributes:**

| QName                            | Type    | Fixed | Default | Use      | Annotation |
|----------------------------------|---------|-------|---------|----------|------------|
| release                          | xs:long |       |         | required |            |
| ANY attribute from ANY namespace |         |       |         |          |            |

**stmt\_value\_index**

The element represents the position of the input parameter marker or host variable used in the SQL statement. See monitor element "stmt\_value\_index - Value index" on page 1259 for more details.

**Contained by:**

**Element content:**

| Type   | Facet |
|--------|-------|
| xs:int |       |

**stmt\_value\_isnull**

**Contained by:**

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

**stmt\_value\_isreopt**

**Contained by:**

**Attributes:**

| QName | Type   | Fixed | Default | Use      | Annotation |
|-------|--------|-------|---------|----------|------------|
| id    | xs:int |       |         | required |            |

**stmt\_value\_type**

“stmt\_value\_type - Value type monitor element” on page 1261

**Contained by:**

**Element content:**

| Type      | Facet          |
|-----------|----------------|
| xs:string | Max length: 16 |

**stmt\_value\_data**

The element contains a string representation of a data value associated with an SQL statement. See monitor element “stmt\_value\_data - Value data” on page 1259 for more details.

**Contained by:**

**Element content:**

| Type      | Facet             |
|-----------|-------------------|
| xs:string | Max length: 32768 |

### **agent\_wait\_time**

See monitor element “agent\_wait\_time - Agent wait time monitor element” on page 654 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **agent\_waits\_total**

See monitor element “agent\_waits\_total - Total agent waits monitor element” on page 655 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **tcpip\_rcv\_wait\_time**

See monitor element “tcpip\_rcv\_wait\_time - TCP/IP received wait time monitor element” on page 1298 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **tcpip\_rcvs\_total**

See monitor element “tcpip\_rcvs\_total - TCP/IP receives total monitor element” on page 1299 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **client\_idle\_wait\_time**

See monitor element “client\_idle\_wait\_time - Client idle wait time monitor element” on page 712 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **ipc\_rcv\_wait\_time**

See monitor element "ipc\_rcv\_wait\_time - Interprocess communication received wait time monitor element" on page 888 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **ipc\_rcvs\_total**

See monitor element "ipc\_rcvs\_total - Interprocess communication receives total monitor element" on page 889 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **ipc\_send\_wait\_time**

See monitor element "ipc\_send\_wait\_time - Interprocess communication send wait time monitor element" on page 891 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **ipc\_sends\_total**

See monitor element "ipc\_sends\_total - Interprocess communication send total monitor element" on page 892 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### tcpip\_send\_wait\_time

See monitor element “tcpip\_send\_wait\_time - TCP/IP send wait time monitor element” on page 1301for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### tcpip\_sends\_total

See monitor element “tcpip\_sends\_total - TCP/IP sends total monitor element” on page 1302for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### total\_wait\_time

See monitor element “total\_wait\_time - Total wait time monitor element” on page 1386for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### total\_rqst\_time

See monitor element “total\_rqst\_time - Total request time monitor element” on page 1362for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **rqsts\_completed\_total**

See monitor element “rqsts\_completed\_total - Total requests completed monitor element” on page 1197for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_app\_rqst\_time**

See monitor element “total\_app\_rqst\_time - Total application request time monitor element” on page 1316for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **app\_rqsts\_completed\_total**

See monitor element “app\_rqsts\_completed\_total - Total application requests completed monitor element” on page 665for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **act\_completed\_total**

See monitor element “act\_completed\_total - Total completed activities monitor element” on page 637for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **tcpip\_send\_volume**

See monitor element “tcpip\_send\_volume - TCP/IP send volume monitor element” on page 1300for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **tcpip\_rcv\_volume**

See monitor element "tcpip\_rcv\_volume - TCP/IP received volume monitor element" on page 1297for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **ipc\_send\_volume**

See monitor element "ipc\_send\_volume - Interprocess communication send volume monitor element" on page 890for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **ipc\_rcv\_volume**

See monitor element "ipc\_rcv\_volume - Interprocess communication received volume monitor element" on page 887for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_rqst\_mapped\_in**

See monitor element "total\_rqst\_mapped\_in - Total request mapped-in monitor element" on page 1362for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_rqst\_mapped\_out**

See monitor element “total\_rqst\_mapped\_out - Total request mapped-out monitor element” on page 1362for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **act\_rejected\_total**

See monitor element “act\_rejected\_total - Total rejected activities monitor element” on page 640for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **act\_aborted\_total**

See monitor element “act\_aborted\_total - Total aborted activities monitor element” on page 636for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_compile\_proc\_time**

See monitor element “total\_compile\_proc\_time - Total compile processing time monitor element” on page 1322for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_compile\_time**

See monitor element “total\_compile\_time - Total compile time monitor element” on page 1323 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_compilations**

See monitor element “total\_compilations - Total compilations monitor element” on page 1321 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_implicit\_compile\_proc\_time**

See monitor element “total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element” on page 1340 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_implicit\_compile\_time**

See monitor element “total\_implicit\_compile\_time - Total implicit compile time monitor element” on page 1341 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_implicit\_compilations**

See monitor element “total\_implicit\_compilations - Total implicit complications monitor element” on page 1339 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_runstats\_proc\_time**

See monitor element "total\_runstats\_proc\_time - Total runtime statistics processing time monitor element" on page 1364 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_runstats\_time**

See monitor element "total\_runstats\_time - Total runtime statistics time monitor element" on page 1365 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_runstats**

See monitor element "total\_runstats - Total runtime statistics monitor element" on page 1363 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_reorg\_proc\_time**

See monitor element "total\_reorg\_proc\_time - Total reorganization processing time monitor element" on page 1350 for more details.

**Contained by:** "system\_metrics" on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_reorg\_time**

See monitor element “total\_reorg\_time - Total reorganization time monitor element” on page 1351 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_reorgs**

See monitor element “total\_reorgs - Total reorganizations monitor element” on page 1352 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_load\_proc\_time**

See monitor element “total\_load\_proc\_time - Total load processing time monitor element” on page 1342 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_load\_time**

See monitor element “total\_load\_time - Total load time monitor element” on page 1343 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_loads**

See monitor element “total\_loads - Total loads monitor element” on page 1344 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_commit\_proc\_time**

See monitor element “total\_commit\_proc\_time - Total commits processing time monitor element” on page 1319 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_commit\_time**

See monitor element “total\_commit\_time - Total commit time monitor element” on page 1320 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_app\_commits**

See monitor element “total\_app\_commits - Total application commits monitor elements” on page 1314 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_rollback\_proc\_time**

See monitor element “total\_rollback\_proc\_time - Total rollback processing time monitor element” on page 1353 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_rollback\_time**

See monitor element “total\_rollback\_time - Total rollback time monitor element” on page 1354for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_app\_rollbacks**

See monitor element “total\_app\_rollbacks - Total application rollbacks monitor element” on page 1315for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **int\_commits**

See monitor element “int\_commits - Internal commits monitor element” on page 880for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **int\_rollbacks**

See monitor element “int\_rollbacks - Internal rollbacks monitor element” on page 882for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **cat\_cache\_inserts**

See monitor element “cat\_cache\_inserts - Catalog cache inserts monitor element” on page 698 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **cat\_cache\_lookups**

See monitor element “cat\_cache\_lookups - Catalog cache lookups monitor element” on page 699 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pkg\_cache\_inserts**

See monitor element “pkg\_cache\_inserts - Package cache inserts monitor element” on page 1020 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pkg\_cache\_lookups**

See monitor element “pkg\_cache\_lookups - Package cache lookups monitor element” on page 1021 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **act\_rqsts\_total**

See monitor element “act\_rqsts\_total - Total activity requests monitor elements” on page 642 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_stats\_fabrication\_proc\_time**

See monitor element “total\_stats\_fabrication\_proc\_time - Total statistics fabrication processing time monitor element” on page 1377 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_stats\_fabrication\_time**

See monitor element “total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_stats\_fabrications**

See monitor element “total\_stats\_fabrications - Total statistics fabrications monitor elements” on page 1379 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_sync\_runstats\_proc\_time**

See monitor element “total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381 for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_sync\_runstats\_time**

See monitor element “total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_sync\_runstats**

See monitor element “total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element” on page 1382for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **app\_act\_completed\_total**

See monitor element “app\_act\_completed\_total - Total successful external coordinator activities monitor element” on page 662for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **app\_act\_aborted\_total**

See monitor element “app\_act\_aborted\_total - Total failed external coordinator activities monitor element” on page 661for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **app\_act\_rejected\_total**

See monitor element “app\_act\_rejected\_total - Total rejected external coordinator activities monitor element” on page 664for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_request\_proc\_time**

See monitor element “total\_connect\_request\_proc\_time - Total connection or switch user request processing time monitor element” on page 1328for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_request\_time**

See monitor element “total\_connect\_request\_time - Total connection or switch user request time monitor element” on page 1330for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_requests**

See monitor element “total\_connect\_requests - Connection or switch user requests monitor element” on page 1329for more details.

**Contained by:** “system\_metrics” on page 280

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_authentication\_proc\_time**

See monitor element “total\_connect\_authentication\_proc\_time - Total connection authentication processing time monitor element” on page 1325 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_authentication\_time**

See monitor element “total\_connect\_authentication\_time - Total connection or switch user authentication request time monitor element” on page 1327 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_connect\_authentications**

See monitor element “total\_connect\_authentications - Connections or switch user authentications performed monitor element” on page 1326 for more details.

**Contained by:** “system\_metrics” on page 280

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **wlm\_queue\_time\_total**

See monitor element “wlm\_queue\_time\_total - Workload manager total queue time monitor element” on page 1422 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### wlm\_queue\_assignments\_total

See monitor element “wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element” on page 1421 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_tq\_rcv\_wait\_time

See monitor element “fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element” on page 840 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_message\_rcv\_wait\_time

See monitor element “fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element” on page 824 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_tq\_send\_wait\_time

See monitor element “fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element” on page 844 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_message\_send\_wait\_time

See monitor element “fcm\_message\_send\_wait\_time - FCM message send wait time monitor element” on page 828 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **lock\_wait\_time**

See monitor element "lock\_wait\_time - Time waited on locks monitor element" on page 924 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **lock\_waits**

See monitor element "lock\_waits - Lock waits monitor element" on page 928 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **direct\_read\_time**

See monitor element "direct\_read\_time - Direct read time monitor element" on page 792 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **direct\_read\_reqs**

See monitor element "direct\_read\_reqs - Direct read requests monitor element" on page 790 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **direct\_write\_time**

See monitor element “direct\_write\_time - Direct write time monitor element” on page 798 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **direct\_write\_reqs**

See monitor element “direct\_write\_reqs - Direct write requests monitor element” on page 796 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **log\_buffer\_wait\_time**

See monitor element “log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **num\_log\_buffer\_full**

See monitor element “num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element” on page 975 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### log\_disk\_wait\_time

See monitor element “log\_disk\_wait\_time - Log disk wait time monitor element” on page 935for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### log\_disk\_waits\_total

See monitor element “log\_disk\_waits\_total - Total log disk waits monitor element” on page 936for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_write\_time

See monitor element “pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_read\_time

See monitor element “pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### audit\_file\_write\_wait\_time

See monitor element “audit\_file\_write\_wait\_time - Audit file write wait time monitor element” on page 680for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **audit\_file\_writes\_total**

See monitor element “audit\_file\_writes\_total - Total audit files written monitor element” on page 682for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **audit\_subsystem\_wait\_time**

See monitor element “audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **audit\_subsystem\_waits\_total**

See monitor element “audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element” on page 685for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **diaglog\_write\_wait\_time**

See monitor element “diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element” on page 788for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **diaglog\_writes\_total**

See monitor element “diaglog\_writes\_total - Total diagnostic log file writes monitor element” on page 789for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_send\_wait\_time**

See monitor element “fcm\_send\_wait\_time - FCM send wait time monitor element” on page 836for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_rcv\_wait\_time**

See monitor element “fcm\_rcv\_wait\_time - FCM received wait time monitor element” on page 832for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_act\_wait\_time**

See monitor element “total\_act\_wait\_time - Total activity wait time monitor element” on page 1312for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_section\_sort\_proc\_time**

See monitor element “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_section\_sort\_time**

See monitor element “total\_section\_sort\_time - Total section sort time monitor element” on page 1370for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_section\_sorts**

See monitor element “total\_section\_sorts - Total section sorts monitor element” on page 1371for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_act\_time**

See monitor element “total\_act\_time - Total activity time monitor element” on page 1311for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **rows\_read**

See monitor element “rows\_read - Rows read monitor element” on page 1191for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **rows\_modified**

See monitor element "rows\_modified - Rows modified monitor element" on page 1190 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_data\_l\_reads**

See monitor element "pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_index\_l\_reads**

See monitor element "pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_temp\_data\_l\_reads**

See monitor element "pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116 for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_temp\_index\_l\_reads**

See monitor element “pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_xda\_l\_reads**

See monitor element “pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_temp\_xda\_l\_reads**

See monitor element “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_cpu\_time**

See monitor element “total\_cpu\_time - Total CPU time monitor element” on page 1330 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_data\_p\_reads

See monitor element “pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_temp\_data\_p\_reads

See monitor element “pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_xda\_p\_reads

See monitor element “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_temp\_xda\_p\_reads

See monitor element “pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_index\_p\_reads

See monitor element “pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_temp\_index\_p\_reads**

See monitor element “pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_data\_writes**

See monitor element “pool\_data\_writes - Buffer pool data writes monitor element” on page 1052for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_xda\_writes**

See monitor element “pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_index\_writes**

See monitor element “pool\_index\_writes - Buffer pool index writes monitor element” on page 1084for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **direct\_reads**

See monitor element “direct\_reads - Direct reads from database monitor element” on page 794 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **direct\_writes**

See monitor element “direct\_writes - Direct writes to database monitor element” on page 800 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **rows\_returned**

See monitor element “rows\_returned - Rows returned monitor element” on page 1193 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **deadlocks**

See monitor element “deadlocks - Deadlocks detected monitor element” on page 783 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_timeouts

See monitor element “lock\_timeouts - Number of lock timeouts monitor element” on page 920for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_escals

See monitor element “lock\_escals - Number of lock escalations monitor element” on page 905for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_sends\_total

See monitor element “fcm\_sends\_total - FCM sends total monitor element” on page 838for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_recvs\_total

See monitor element “fcm\_recvs\_total - FCM receives total monitor element” on page 834for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### fcm\_send\_volume

See monitor element “fcm\_send\_volume - FCM send volume monitor element” on page 835for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **fcm\_recv\_volume**

See monitor element "fcm\_recv\_volume - FCM received volume monitor element" on page 831for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **fcm\_message\_sends\_total**

See monitor element "fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **fcm\_message\_recvs\_total**

See monitor element "fcm\_message\_recvs\_total - Total FCM message receives monitor element" on page 826for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **fcm\_message\_send\_volume**

See monitor element "fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_message\_rcv\_volume**

See monitor element “fcm\_message\_rcv\_volume - FCM message received volume monitor element” on page 823for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_tq\_sends\_total**

See monitor element “fcm\_tq\_sends\_total - FCM table queue send total monitor element” on page 846for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_tq\_recvs\_total**

See monitor element “fcm\_tq\_recvs\_total - FCM table queue receives total monitor element” on page 842for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_tq\_send\_volume**

See monitor element “fcm\_tq\_send\_volume - FCM table queue send volume monitor element” on page 843for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **fcm\_tq\_rcv\_volume**

See monitor element “fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element” on page 839 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **tq\_tot\_send\_spills**

See monitor element “tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element” on page 1396 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **post\_threshold\_sorts**

See monitor element “post\_threshold\_sorts - Post threshold sorts monitor element” on page 1150 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **post\_shrthreshold\_sorts**

See monitor element “post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **sort\_overflows**

See monitor element “sort\_overflows - Sort overflows monitor element” on page 1224 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **audit\_events\_total**

See monitor element “audit\_events\_total - Total audit events monitor element” on page 679for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_sorts**

See monitor element “total\_sorts - Total sorts monitor element” on page 1375for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **stmt\_exec\_time**

See monitor element “stmt\_exec\_time - Statement execution time monitor element” on page 1245for more details.

**Contained by:** “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **coord\_stmt\_exec\_time**

See monitor element “coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element” on page 751for more details.

**Contained by:** “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_routine\_non\_sect\_proc\_time**

See monitor element “total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element” on page 1356for more details.

**Contained by:** “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_routine\_non\_sect\_time**

See monitor element “total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements” on page 1356for more details.

**Contained by:** “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_section\_proc\_time**

See monitor element “total\_section\_proc\_time - Total section processing time monitor element” on page 1367for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**total\_section\_time**

See monitor element “total\_section\_time - Total section time monitor element” on page 1373for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_app\_section\_executions**

See monitor element “total\_app\_section\_executions - Total application section executions monitor element” on page 1317for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_routine\_user\_code\_proc\_time**

See monitor element “total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element” on page 1359for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_routine\_user\_code\_time**

See monitor element “total\_routine\_user\_code\_time - Total routine user code time monitor element” on page 1360for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **total\_routine\_time**

See monitor element “total\_routine\_time - Total routine time monitor element” on page 1357for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **thresh\_violations**

See monitor element “thresh\_violations - Number of threshold violations monitor element” on page 1304for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **num\_lw\_thresh\_exceeded**

See monitor element "num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element" on page 979for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_routine\_invocations**

See monitor element "total\_routine\_invocations - Total routine invocations monitor elements" on page 1355for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **lock\_wait\_time\_global**

See monitor element "lock\_wait\_time\_global - Lock wait time global monitor element" on page 926for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **lock\_waits\_global**

See monitor element "lock\_waits\_global - Lock waits global monitor element" on page 930for more details.

**Contained by:** "system\_metrics" on page 280 "activity\_metrics" on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **reclaim\_wait\_time**

See monitor element “reclaim\_wait\_time - Reclaim wait time monitor element” on page 1170 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **spacemappage\_reclaim\_wait\_time**

See monitor element “spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element” on page 1236 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **lock\_timeouts\_global**

See monitor element “lock\_timeouts\_global - Lock timeouts global monitor element” on page 922 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **lock\_escals\_maxlocks**

See monitor element “lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element” on page 910 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_escals\_locklist

See monitor element “lock\_escals\_locklist - Number of locklist lock escalations monitor element” on page 909 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### lock\_escals\_global

See monitor element “lock\_escals\_global - Number of global lock escalations monitor element” on page 907 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### cf\_wait\_time

See monitor element “cf\_wait\_time - cluster caching facility wait time monitor element” on page 703 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### cf\_waits

See monitor element “cf\_waits - Number of cluster caching facility waits monitor element” on page 703 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_data\_gbp\_l\_reads

See monitor element “pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element” on page 1044 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_data\_gbp\_p\_reads**

See monitor element “pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element” on page 1045 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_data\_lbp\_pages\_found**

See monitor element “pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element” on page 1047 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_data\_gbp\_invalid\_pages**

See monitor element “pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element” on page 1042 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_index\_gbp\_l\_reads**

See monitor element “pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element” on page 1075 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_index\_gbp\_p\_reads

See monitor element “pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements” on page 1077 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_index\_lbp\_pages\_found

See monitor element “pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element” on page 1078 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_index\_gbp\_invalid\_pages

See monitor element “pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element” on page 1074 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_xda\_gbp\_l\_reads

See monitor element “pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element” on page 1132 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_xda\_gbp\_p\_reads**

See monitor element “pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element” on page 1134 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_xda\_lbp\_pages\_found**

See monitor element “pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element” on page 1138 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_xda\_gbp\_invalid\_pages**

See monitor element “pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element” on page 1131 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **evmon\_wait\_time**

See monitor element “evmon\_wait\_time - Event monitor wait time monitor element” on page 814 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **evmon\_waits\_total**

See monitor element “evmon\_waits\_total - Event monitor total waits monitor element” on page 816 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_extended\_latch\_wait\_time**

See monitor element “total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_extended\_latch\_waits**

See monitor element “total\_extended\_latch\_waits - Total extended latch waits monitor element” on page 1336 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **total\_disp\_run\_queue\_time**

See monitor element “total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

#### **pool\_queued\_async\_data\_reqs**

See monitor element “pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element” on page 1089 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_index\_reqs

See monitor element “pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element” on page 1093 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_xda\_reqs

See monitor element “pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element” on page 1111 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_data\_reqs

See monitor element “pool\_queued\_async\_temp\_data\_reqs - Data prefetch requests for temporary table spaces monitor element” on page 1099 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_index\_reqs

See monitor element “pool\_queued\_async\_temp\_index\_reqs - Index prefetch requests for temporary table spaces monitor element” on page 1103 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_xda\_reqs

See monitor element “pool\_queued\_async\_temp\_xda\_reqs - XDA data prefetch requests for temporary table spaces monitor element” on page 1107for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_other\_reqs

See monitor element “pool\_queued\_async\_other\_reqs - Other requests handled by prefetchers monitor element” on page 1095for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_data\_pages

See monitor element “pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element” on page 1087for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_index\_pages

See monitor element “pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element” on page 1091for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_xda\_pages

See monitor element “pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element” on page 1109 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_data\_pages

See monitor element “pool\_queued\_async\_temp\_data\_pages - Data pages prefetch requests for temporary table spaces monitor element” on page 1097 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_index\_pages

See monitor element “pool\_queued\_async\_temp\_index\_pages - Index pages prefetch requests for temporary table spaces monitor element” on page 1101 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_queued\_async\_temp\_xda\_pages

See monitor element “pool\_queued\_async\_temp\_xda\_pages - XDA data pages prefetch requests for temporary table spaces monitor element” on page 1105 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_failed\_async\_data\_reqs**

See monitor element “pool\_failed\_async\_data\_reqs - Failed data prefetch requests monitor element” on page 1057 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_failed\_async\_index\_reqs**

See monitor element “pool\_failed\_async\_index\_reqs - Failed index prefetch requests monitor element” on page 1059 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_failed\_async\_xda\_reqs**

See monitor element “pool\_failed\_async\_xda\_reqs - Failed XDA prefetch requests monitor element” on page 1069 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### **pool\_failed\_async\_temp\_data\_reqs**

See monitor element “pool\_failed\_async\_temp\_data\_reqs - Failed data prefetch requests for temporary table spaces monitor element” on page 1063 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### **Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_failed\_async\_temp\_index\_reqs

See monitor element “pool\_failed\_async\_temp\_index\_reqs - Failed index prefetch requests for temporary table spaces monitor element” on page 1065 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_failed\_async\_temp\_xda\_reqs

See monitor element “pool\_failed\_async\_temp\_xda\_reqs - Failed XDA prefetch requests for temporary table spaces monitor element” on page 1067 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_failed\_async\_other\_reqs

See monitor element “pool\_failed\_async\_other\_reqs - Failed non-prefetch requests monitor element” on page 1061 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### total\_peds

See monitor element “total\_peds - Total partial early distincts monitor element” on page 1349 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

#### Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### disabled\_peds

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### post\_threshold\_peds

See monitor element “post\_threshold\_peds - Partial early distincts threshold monitor element” on page 1148 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### total\_peas

See monitor element “total\_peas - Total partial early aggregations monitor element” on page 1346 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### post\_threshold\_peas

See monitor element “post\_threshold\_peas - Partial early aggregation threshold monitor element” on page 1146 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### tq\_sort\_heap\_requests

See monitor element “tq\_sort\_heap\_requests - Table queue sort heap requests monitor element” on page 1394 for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**tq\_sort\_heap\_rejections**

See monitor element “tq\_sort\_heap\_rejections - Table queue sort heap rejections monitor element” on page 1392for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**prefetch\_wait\_time**

See monitor element “prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**prefetch\_waits**

See monitor element “prefetch\_waits - Prefetcher wait count monitor element” on page 1153for more details.

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

**pool\_data\_gbp\_indep\_pages\_found\_in\_lbp**

**Contained by:** “system\_metrics” on page 280 “activity\_metrics” on page 282

**Element content:**

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_index\_gbp\_indep\_pages\_found\_in\_lbp

Contained by: “system\_metrics” on page 280 “activity\_metrics” on page 282

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

### pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp

Contained by: “system\_metrics” on page 280 “activity\_metrics” on page 282

Element content:

| Type    | Facet |
|---------|-------|
| xs:long |       |

## Database event monitoring

### Data generated by database event monitors

Database event monitors produce data about database-level counters. You can choose to have the output from a database event monitor to regular tables, files or pipes.

Regardless of the output format you choose, all database event data comes from one of two logical groups:

- “event\_db logical data group” on page 54
- “event\_dbmemuse logical data group” on page 58

In addition, if you choose to have the database event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a database event monitor:

Information written by an database event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for the database event monitor, by default, three tables are produced, each containing monitor elements from one or more logical data groups:

*Table 60. Tables produced by DATABASE write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name          | Logical data groups reported |
|-----------------------------|------------------------------|
| DB_ <i>evmon-name</i>       | event_db                     |
| DBMEMUSE_ <i>evmon-name</i> | event_dbmemuse               |

Table 60. Tables produced by DATABASE write-to-table event monitors (continued). The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name         | Logical data groups reported  |
|----------------------------|---|
| CONTROL_ <i>evmon-name</i> | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

### Tables produced

Table 61. Information returned for a database event monitor: Default table name: *DB\_evmon-name*

| Column name            | Data type   | Description   |
|------------------------|-------------|---|
| APPL_SECTION_INSERTS   | BIGINT      | appl_section_inserts - Section inserts                          |
| APPL_SECTION_LOOKUPS   | BIGINT      | appl_section_lookups - Section lookups                          |
| ASYNC_RUNSTATS         | BIGINT      | async_runstats - Total number of asynchronous RUNSTATS requests |
| BINDS_PRECOMPILES      | BIGINT      | binds_precompiles - Binds/precompiles attempted                 |
| BLOCKS_PENDING_CLEANUP | BIGINT      | blocks_pending_cleanup - Pending cleanup rolled-out blocks      |
| CAT_CACHE_HEAP_FULL    | BIGINT      |   |
| CAT_CACHE_INSERTS      | BIGINT      | cat_cache_inserts - Catalog cache inserts                       |
| CAT_CACHE_LOOKUPS      | BIGINT      | cat_cache_lookups - Catalog cache lookups                       |
| CAT_CACHE_OVERFLOWS    | BIGINT      | cat_cache_overflows - Catalog cache overflows                   |
| CAT_CACHE_SIZE_TOP     | BIGINT      | cat_cache_size_top - Catalog cache high watermark               |
| CATALOG_NODE           | BIGINT      | catalog_node - Catalog node number                              |
| CATALOG_NODE_NAME      | VARCHAR(32) | catalog_node_name - Catalog node network name                   |
| COMMIT_SQL_STMTS       | BIGINT      | commit_sql_stmts - Commit statements attempted                  |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name               | Data type | Description   |
|---------------------------|-----------|---|
| CONNECTIONS_TOP           | BIGINT    | connections_top - Maximum number of concurrent connections    |
| DB_HEAP_TOP               | BIGINT    | db_heap_top - Maximum database heap allocated                 |
| DDL_SQL_STMTS             | BIGINT    | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| DEADLOCKS                 | BIGINT    | deadlocks - Deadlocks detected                                |
| DIRECT_READ_REQS          | BIGINT    | direct_read_reqs - Direct read requests                       |
| DIRECT_READ_TIME          | BIGINT    | direct_read_time - Direct read time                           |
| DIRECT_READS              | BIGINT    | direct_reads - Direct reads from database                     |
| DIRECT_WRITE_REQS         | BIGINT    | direct_write_reqs - Direct write requests                     |
| DIRECT_WRITE_TIME         | BIGINT    | direct_write_time - Direct write time                         |
| DIRECT_WRITES             | BIGINT    | direct_writes - Direct writes to database                     |
| DISCONN_TIME              | TIMESTAMP | disconn_time - Database deactivation timestamp                |
| DYNAMIC_SQL_STMTS         | BIGINT    | dynamic_sql_stmts - Dynamic SQL statements attempted          |
| ELAPSED_EXEC_TIME         | BIGINT    | elapsed_exec_time - Statement execution elapsed time          |
| EVMON_ACTIVATES           | BIGINT    | evmon_activates - Number of event monitor activations         |
| EVMON_FLUSHES             | BIGINT    | evmon_flushes - Number of event monitor flushes               |
| FAILED_SQL_STMTS          | BIGINT    | failed_sql_stmts - Failed statement operations                |
| FILES_CLOSED              | BIGINT    | files_closed - Database files closed                          |
| HASH_JOIN_OVERFLOWS       | BIGINT    | hash_join_overflows - Hash join overflows                     |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT    | hash_join_small_overflows - Hash join small overflows         |
| INT_AUTO_REBINDS          | BIGINT    | int_auto_rebinds - Internal automatic rebinds                 |
| INT_COMMITS               | BIGINT    | int_commits - Internal commits                                |
| INT_ROLLBACKS             | BIGINT    | int_rollback - Internal rollbacks                             |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name              | Data type | Description  |
|--------------------------|-----------|--|
| INT_ROWS_DELETED         | BIGINT    | int_rows_deleted - Internal rows deleted                                   |
| INT_ROWS_INSERTED        | BIGINT    | int_rows_inserted - Internal rows inserted                                 |
| INT_ROWS_UPDATED         | BIGINT    | int_rows_updated - Internal rows updated                                   |
| LOCK_ESCALS              | BIGINT    | lock_escals - Number of lock escalations                                   |
| LOCK_TIMEOUTS            | BIGINT    | lock_timeouts - Number of lock timeouts                                    |
| LOCK_WAIT_TIME           | BIGINT    | lock_wait_time - Time waited on locks                                      |
| LOCK_WAITS               | BIGINT    | lock_waits - Lock waits  |
| LOG_FILE_ARCHIVE         | BIGINT    |  |
| LOG_FILE_NUM_CURR        | BIGINT    |  |
| LOG_FILE_NUM_FIRST       | BIGINT    |  |
| LOG_FILE_NUM_LAST        | BIGINT    |  |
| LOG_HELD_BY_DIRTY_PAGES  | BIGINT    | log_held_by_dirty_pages - Amount of log space accounted for by dirty pages |
| LOG_READ_TIME            | BIGINT    | log_read_time - Log read time  |
| LOG_READS                | BIGINT    | log_reads - Number of log pages read                                       |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT    | log_to_redo_for_recovery - Amount of log to be redone for recovery         |
| LOG_WRITE_TIME           | BIGINT    | log_write_time - Log write time  |
| LOG_WRITES               | BIGINT    | log_writes - Number of log pages written                                   |
| NUM_LOG_BUFF_FULL        | BIGINT    |  |
| NUM_LOG_DATA_IN_BUFF     | BIGINT    |  |
| NUM_LOG_PART_PAGE_IO     | BIGINT    | num_log_part_page_io - Number of partial log page writes                   |
| NUM_LOG_READ_IO          | BIGINT    | num_log_read_io - Number of log reads                                      |
| NUM_LOG_WRITE_IO         | BIGINT    | num_log_write_io - Number of log writes                                    |
| NUM_THRESHOLD_VIOLATIONS | INTEGER   | num_threshold_violations - Number of threshold violations                  |
| OLAP_FUNC_OVERFLOWS      | BIGINT    | olap_func_overflows - OLAP function overflows                              |
| PARTIAL_RECORD           | SMALLINT  | partial_record - Partial record  |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name                | Data type | Description   |
|----------------------------|-----------|---|
| PKG_CACHE_INSERTS          | BIGINT    | pkg_cache_inserts - Package cache inserts                                 |
| PKG_CACHE_LOOKUPS          | BIGINT    | pkg_cache_lookups - Package cache lookups                                 |
| PKG_CACHE_NUM_OVERFLOWS    | BIGINT    | pkg_cache_num_overflows - Package cache overflows                         |
| PKG_CACHE_SIZE_TOP         | BIGINT    | pkg_cache_size_top - Package cache high watermark                         |
| POOL_ASYNC_DATA_READ_REQS  | BIGINT    | pool_async_data_read_reqs - Buffer pool asynchronous read requests        |
| POOL_ASYNC_DATA_READS      | BIGINT    | pool_async_data_reads - Buffer pool asynchronous data reads               |
| POOL_ASYNC_DATA_WRITES     | BIGINT    | pool_async_data_writes - Buffer pool asynchronous data writes             |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT    | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_INDEX_READS     | BIGINT    | pool_async_index_reads - Buffer pool asynchronous index reads             |
| POOL_ASYNC_INDEX_WRITES    | BIGINT    | pool_async_index_writes - Buffer pool asynchronous index writes           |
| POOL_ASYNC_READ_TIME       | BIGINT    | pool_async_read_time - Buffer pool asynchronous read time                 |
| POOL_ASYNC_WRITE_TIME      | BIGINT    | pool_async_write_time - Buffer pool asynchronous write time               |
| POOL_ASYNC_XDA_READ_REQS   | BIGINT    | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests     |
| POOL_ASYNC_XDA_READS       | BIGINT    | pool_async_xda_reads - Buffer pool asynchronous XDA data reads            |
| POOL_ASYNC_XDA_WRITES      | BIGINT    | pool_async_xda_writes - Buffer pool asynchronous XDA data writes          |
| POOL_DATA_L_READS          | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                        |
| POOL_DATA_P_READS          | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                       |
| POOL_DATA_WRITES           | BIGINT    | pool_data_writes - Buffer pool data writes                                |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name                  | Data type | Description   |
|------------------------------|-----------|---|
| POOL_DRTY_PG_STEAL_CLNS      | BIGINT    | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered  |
| POOL_DRTY_PG_THRSH_CLNS      | BIGINT    | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered    |
| POOL_INDEX_L_READS           | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_INDEX_P_READS           | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_INDEX_WRITES            | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| POOL_LSN_GAP_CLNS            | BIGINT    | pool_lsn_gap_clns - Buffer pool log space cleaners triggered          |
| POOL_NO_VICTIM_BUFFER        | BIGINT    | pool_no_victim_buffer - Buffer pool no victim buffers                 |
| POOL_READ_TIME               | BIGINT    | pool_read_time - Total buffer pool physical read time                 |
| POOL_TEMP_DATA_L_READS       | BIGINT    | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_DATA_P_READS       | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_TEMP_INDEX_L_READS      | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_TEMP_INDEX_P_READS      | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_TEMP_XDA_L_READS        | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| POOL_TEMP_XDA_P_READS        | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_WRITE_TIME              | BIGINT    | pool_write_time - Total buffer pool physical write time               |
| POOL_XDA_L_READS             | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_XDA_P_READS             | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_XDA_WRITES              | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT    | post_shrthreshold_hash_joins - Post threshold hash joins              |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name             | Data type | Description   |
|-------------------------|-----------|---|
| POST_SHRTHRESHOLD_SORTS | BIGINT    | post_shrthreshold_sorts - Post shared threshold sorts   |
| PREFETCH_WAIT_TIME      | BIGINT    | prefetch_wait_time - Time waited for prefetch   |
| ROLLBACK_SQL_STMTS      | BIGINT    | rollback_sql_stmts - Rollback statements attempted  |
| ROWS_DELETED            | BIGINT    | rows_deleted - Rows deleted   |
| ROWS_INSERTED           | BIGINT    | rows_inserted - Rows inserted   |
| ROWS_READ               | BIGINT    | rows_read - Rows read   |
| ROWS_SELECTED           | BIGINT    | rows_selected - Rows selected   |
| ROWS_UPDATED            | BIGINT    | rows_updated - Rows updated   |
| SEC_LOG_USED_TOP        | BIGINT    | sec_log_used_top - Maximum secondary log space used   |
| SELECT_SQL_STMTS        | BIGINT    | select_sql_stmts - Select SQL statements executed   |
| SERVER_PLATFORM         | INTEGER   | server_platform - Server operating system   |
| SORT_OVERFLOWS          | BIGINT    | sort_overflows - Sort overflows   |
| SORT_SHRHEAP_TOP        | BIGINT    | sort_shrheap_top - Sort share heap high watermark   |
| STATIC_SQL_STMTS        | BIGINT    | static_sql_stmts - Static SQL statements attempted  |
| STATS_CACHE_SIZE        | BIGINT    | stats_cache_size - Size of statistics cache   |
| STATS_FABRICATE_TIME    | BIGINT    | stats_fabricate_time - Total time spent on statistics fabrication activities                            |
| STATS_FABRICATIONS      | BIGINT    | stats_fabrications - Total number of statistics fabrications  |
| SYNC_RUNSTATS           | BIGINT    | sync_runstats - Total number of synchronous RUNSTATS activities   |
| SYNC_RUNSTATS_TIME      | BIGINT    | "sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 1266 |
| TOT_LOG_USED_TOP        | BIGINT    | tot_log_used_top - Maximum total log space used   |
| TOTAL_CONS              | BIGINT    | total_cons - Connects since database activation   |
| TOTAL_HASH_JOINS        | BIGINT    | total_hash_joins - Total hash joins   |

Table 61. Information returned for a database event monitor: Default table name: DB\_evmon-name (continued)

| Column name           | Data type | Description  |
|-----------------------|-----------|--|
| TOTAL_HASH_LOOPS      | BIGINT    | total_hash_loops - Total hash loops                          |
| TOTAL_OLAP_FUNCS      | BIGINT    | total_olap_funcs - Total OLAP functions                      |
| TOTAL_SORT_TIME       | BIGINT    | total_sort_time - Total sort time                            |
| TOTAL_SORTS           | BIGINT    | total_sorts - Total sorts                                    |
| UID_SQL_STMTS         | BIGINT    | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| UNREAD_PREFETCH_PAGES | BIGINT    | unread_prefetch_pages - Unread prefetch pages                |
| X_LOCK_ESCALS         | BIGINT    | x_lock_escalations - Exclusive lock escalations              |
| XQUERY_STMTS          | BIGINT    | xquery_stmts - XQuery Statements Attempted                   |

Table 62. Information returned for a database event monitor: Default table name: DBMEMUSE\_evmon-name

| Column name       | Data type     | Description   |
|-------------------|---------------|---|
| EVMON_ACTIVATES   | BIGINT        | evmon_activates - Number of event monitor activations |
| EVMON_FLUSHES     | BIGINT        | evmon_flushes - Number of event monitor flushes       |
| POOL_CUR_SIZE     | BIGINT        | pool_cur_size - Current size of memory pool           |
| POOL_ID           | BIGINT        | pool_id - Memory pool identifier                      |
| POOL_MAX_SIZE     | BIGINT        |   |
| POOL_SECONDARY_ID | CHARACTER(32) | pool_secondary_id - Memory pool secondary identifier  |
| POOL_WATERMARK    | BIGINT        | pool_watermark - Memory pool watermark                |

Table 63. Information returned for a database event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Threshold violation event monitoring

### Data generated by threshold violation event monitors

Threshold violation event monitors produce data about threshold violations. You can choose to have the output from a database event monitor to regular tables, files or pipes.

Regardless of the output format you choose, all threshold violation event data comes from the event\_thresholdviolations logical data group. In addition, if you choose to have the event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a threshold violation event monitor:

Information written by a threshold violations event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a threshold violations event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

*Table 64. Tables produced by THRESHOLD write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name                     | Logical data groups reported  |
|--|---|
| THRESHOLDVIOLATIONS_ <i>evmon-name</i> | event_thresholdviolations   |
| CONTROL_ <i>evmon-name</i>             | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

### Tables produced

*Table 65. Information returned for a threshold violations event monitor: Default table name: THRESHOLDVIOLATIONS\_*evmon-name**

| Column Name        | Data Type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | “partition_key - Partitioning key monitor element” on page 1016 |
| ACTIVATE_TIMESTAMP | TIMESTAMP    | activate_timestamp - Activate timestamp                         |
| ACTIVITY_COLLECTED | CHARACTER(1) | activity_collected - Activity collected                         |
| ACTIVITY_ID        | BIGINT       | activity_id - Activity ID                                       |
| AGENT_ID           | BIGINT       | agent_id - Application handle (agent ID)                        |
| APPL_ID            | VARCHAR(64)  | appl_id - Application ID  |

Table 65. Information returned for a threshold violations event monitor: Default table name: THRESHOLDVIOLATIONS\_evmon-name (continued)

| Column Name                  | Data Type   | Description   |
|------------------------------|-------------|---|
| COORD_PARTITION_NUM          | INTEGER     | coord_partition_num - Coordinator partition number  |
| DESTINATION_SERVICE_CLASS_ID | INTEGER     | “destination_service_class_id – Destination service class ID monitor element” on page 787 |
| PARTITION_NUMBER             | SMALLINT    | partition_number - Partition number   |
| SOURCE_SERVICE_CLASS_ID      | INTEGER     | source_service_class_id - Source service class ID   |
| THRESHOLD_ACTION             | VARCHAR(16) | threshold_action - Threshold action   |
| THRESHOLD_MAXVALUE           | BIGINT      | threshold_maxvalue - Threshold maximum value  |
| THRESHOLD_PREDICATE          | VARCHAR(64) | threshold_predicate - Threshold predicate   |
| THRESHOLD_QUEUESIZE          | BIGINT      | threshold_queuesize - Threshold queue size  |
| THRESHOLDID                  | INTEGER     | thresholdid - Threshold ID  |
| TIME_OF_VIOLATION            | TIMESTAMP   | time_of_violation - Time of violation   |
| UOW_ID                       | INTEGER     | uow_id - Unit of work ID  |

Table 66. Information returned for a threshold violations event monitor: Table name: CONTROL\_evmon-name

| Column Name        | Data Type    | Description   |
|--------------------|--------------|---|
| PARTITION_KEY      | INTEGER      | “partition_key - Partitioning key monitor element” on page 1016 |
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name                         |
| MESSAGE            | VARCHAR(128) | message - Control table message                                 |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message                  |
| PARTITION_NUMBER   | SMALLINT     | partition_number - Partition number                             |

## Statement event monitoring

### Data generated by statement event monitors

Statement event monitors produce data about statements that run on the system. You can choose to have the output from a database event monitor to regular tables, files or pipes.

Regardless of the output format you choose, all statement event data comes from one of three logical groups:

- “event\_stmt logical data group” on page 63
- “event\_connheader logical data group” on page 53
- “event\_subsection logical data group” on page 65

In addition, if you choose to have statement event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a statement event monitor:

Information written by a statement event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a statement event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

*Table 67. Tables produced by STATEMENT write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name            | Logical data groups reported  |
|-------------------------------|---|
| STMT_ <i>evmon-name</i>       | “event_stmt logical data group” on page 63  |
| CONNHEADER_ <i>evmon-name</i> | “event_connheader logical data group” on page 53  |
| CONNHEADER_ <i>evmon-name</i> | “event_connheader logical data group” on page 53  |
| SUBSECTION_ <i>evmon-name</i> | “event_subsection logical data group” on page 65 (only generated in partitioned database environments)  |
| CONTROL_ <i>evmon-name</i>    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

### Tables produced

*Table 68. Information returned for a statement event monitor: Default table name: STMT\_*evmon-name**

| Column name       | Data type    | Description                                     |
|-------------------|--------------|---|
| AGENT_ID          | BIGINT       | agent_id - Application handle (agent ID)        |
| AGENTS_TOP        | BIGINT       | agents_top - Number of agents created           |
| APPL_ID           | VARCHAR(64)  | appl_id - Application ID                        |
| BLOCKING_CURSOR   | SMALLINT     | blocking_cursor - Blocking cursor               |
| CONSISTENCY_TOKEN | CHARACTER    | consistency_token - Package consistency token   |
| CREATOR           | VARCHAR(128) | creator - Application creator                   |
| CURSOR_NAME       | VARCHAR(18)  | cursor_name - Cursor name                       |
| EVMON_FLUSHES     | BIGINT       | evmon_flushes - Number of event monitor flushes |
| FETCH_COUNT       | BIGINT       | fetch_count - Number of successful fetches      |
| INT_ROWS_DELETED  | BIGINT       | int_rows_deleted - Internal rows deleted        |

Table 68. Information returned for a statement event monitor: Default table name: STMT\_evmon-name (continued)

| Column name             | Data type    | Description   |
|-------------------------|--------------|---|
| INT_ROWS_INSERTED       | BIGINT       | int_rows_inserted - Internal rows inserted                            |
| INT_ROWS_UPDATED        | BIGINT       | int_rows_updated - Internal rows updated                              |
| PACKAGE_NAME            | VARCHAR(128) | package_name - Package name   |
| PACKAGE_VERSION_ID      | VARCHAR(64)  | package_version_id - Package version                                  |
| PARTIAL_RECORD          | SMALLINT     | partial_record - Partial record                                       |
| POOL_DATA_L_READS       | BIGINT       | pool_data_l_reads - Buffer pool data logical reads                    |
| POOL_DATA_P_READS       | BIGINT       | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_INDEX_L_READS      | BIGINT       | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_INDEX_P_READS      | BIGINT       | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_TEMP_DATA_L_READS  | BIGINT       | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_DATA_P_READS  | BIGINT       | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_TEMP_INDEX_L_READS | BIGINT       | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_TEMP_INDEX_P_READS | BIGINT       | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_TEMP_XDA_L_READS   | BIGINT       | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| POOL_TEMP_XDA_P_READS   | BIGINT       | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_XDA_L_READS        | BIGINT       | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_XDA_P_READS        | BIGINT       | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| ROWS_READ               | BIGINT       | rows_read - Rows read   |
| ROWS_WRITTEN            | BIGINT       | rows_written - Rows written   |
| SECTION_NUMBER          | BIGINT       | section_number - Section number                                       |
| SEQUENCE_NO             | CHARACTER    | sequence_no - Sequence number   |
| SORT_OVERFLOW           | BIGINT       | sort_overflows - Sort overflows                                       |

Table 68. Information returned for a statement event monitor: Default table name: STMT\_evmon-name (continued)

| Column name          | Data type   | Description  |
|----------------------|-------------|--|
| SQL_REQ_ID           | BIGINT      | sql_req_id - Request identifier for SQL statement                            |
| SQLCABC              | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLCAID              | CHARACTER   | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLCODE              | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD1             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD2             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD3             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD4             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD5             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRD6             | INTEGER     | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRM              | VARCHAR(72) | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLERRP              | CHARACTER   | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLSTATE             | CHARACTER   | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| SQLWARN              | CHARACTER   | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> .     |
| START_TIME           | TIMESTAMP   | start_time - Event start time  |
| STATS_FABRICATE_TIME | BIGINT      | stats_fabricate_time - Total time spent on statistics fabrication activities |
| STMT_OPERATION       | BIGINT      |  |
| STMT_TYPE            | BIGINT      | stmt_type - Statement type   |
| STOP_TIME            | TIMESTAMP   | stop_time - Event stop time  |

Table 68. Information returned for a statement event monitor: Default table name: STMT\_evmon-name (continued)

| Column name        | Data type      | Description   |
|--------------------|----------------|---|
| SYNC_RUNSTATS_TIME | BIGINT         | “sync_runstats_time – Total time spent on synchronous RUNSTATS activities monitor element” on page 1266 |
| SYSTEM_CPU_TIME    | BIGINT         | system_cpu_time - System CPU time   |
| TOTAL_SORT_TIME    | BIGINT         | total_sort_time - Total sort time   |
| TOTAL_SORTS        | BIGINT         | total_sorts - Total sorts   |
| USER_CPU_TIME      | BIGINT         | user_cpu_time - User CPU time   |
| STMT_TEXT          | CLOB (2097152) | stmt_text - SQL statement text  |

Table 69. Information returned for a statement event monitor: Default table name: CONNHEADER\_evmon-name

| Column name     | Data type    | Description  |
|-----------------|--------------|--|
| AGENT_ID        | BIGINT       | agent_id - Application handle (agent ID)                 |
| APPL_ID         | VARCHAR(64)  | appl_id - Application ID                                 |
| APPL_NAME       | VARCHAR(255) | appl_name - Application name                             |
| AUTH_ID         | VARCHAR(128) | auth_id - Authorization ID                               |
| CLIENT_DB_ALIAS | CHARACTER    | client_db_alias - Database alias used by application     |
| CLIENT_NNAME    | VARCHAR(20)  | “client_nname - Client name monitor element” on page 713 |
| CLIENT_PID      | BIGINT       | client_pid - Client process ID                           |
| CLIENT_PLATFORM | INTEGER      | client_platform - Client operating platform              |
| CLIENT_PRDID    | VARCHAR(20)  | client_prdid - Client product and version ID             |
| CLIENT_PROTOCOL | INTEGER      | client_protocol - Client communication protocol          |
| CODEPAGE_ID     | INTEGER      | codepage_id - ID of code page used by application        |
| CONN_TIME       | TIMESTAMP    | conn_time - Time of database connection                  |
| CORR_TOKEN      | VARCHAR(64)  | corr_token - DRDA® correlation token                     |
| EXECUTION_ID    | VARCHAR(128) | execution_id - User login ID                             |
| SEQUENCE_NO     | CHARACTER    | sequence_no - Sequence number                            |
| TERRITORY_CODE  | INTEGER      | territory_code - Database territory code                 |

Table 70. Information returned for a statement event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Table event monitoring

### Data generated by table event monitors

Table event monitors produce aggregate metrics for tables in the database. You can choose to have the output from a database event monitor to regular tables, files or pipes.

**Note:** If you want more detailed usage information for a specific table object, consider creating a usage list for that object.

Regardless of the output format you choose, all table event data comes from the event\_table logical data group. In addition, if you choose to have statement event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a table event monitor:

Information written by an table event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a table event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

Table 71. Tables produced by TABLE write-to-table event monitors. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by evmon-name in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name | Logical data groups reported  |
|--------------------|---|
| TABLE_evmon-name   | event_table   |
| CONTROL_evmon-name | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

## Tables produced

Table 72. Information returned for a table event monitor: Default table name: *TABLE\_evmon-name*

| Column name        | Data type    | Description   |
|--------------------|--------------|---|
| DATA_OBJECT_PAGES  | BIGINT       | data_object_pages - Data object pages                 |
| DATA_PARTITION_ID  | INTEGER      | data_partition_id - Data partition identifier         |
| EVENT_TIME         | TIMESTAMP    | event_time - Event time                               |
| EVMON_ACTIVATES    | BIGINT       | evmon_activates - Number of event monitor activations |
| EVMON_FLUSHES      | BIGINT       | evmon_flushes - Number of event monitor flushes       |
| INDEX_OBJECT_PAGES | BIGINT       | index_object_pages - Index object pages               |
| LOB_OBJECT_PAGES   | BIGINT       | lob_object_pages - LOB object pages                   |
| LONG_OBJECT_PAGES  | BIGINT       | long_object_pages - Long object pages                 |
| OVERFLOW_ACCESSES  | BIGINT       | overflow_accesses - Accesses to overflowed records    |
| PAGE_REORGS        | BIGINT       | page_reorgs - Page reorganizations                    |
| PARTIAL_RECORD     | SMALLINT     | partial_record - Partial record                       |
| ROWS_READ          | BIGINT       | rows_read - Rows read                                 |
| ROWS_WRITTEN       | BIGINT       | rows_written - Rows written                           |
| TABLE_NAME         | VARCHAR(128) | table_name - Default table name                       |
| TABLE_SCHEMA       | VARCHAR(128) | table_schema - Table schema name                      |
| TABLE_TYPE         | BIGINT       | table_type - Table type                               |
| TABLESPACE_ID      | BIGINT       | tablespace_id - Table space identification            |
| XDA_OBJECT_PAGES   | BIGINT       | xda_object_pages - XDA Object Pages                   |

Table 73. Information returned for a table event monitor: Default table name: *CONTROL\_evmon-name*

| Column name        | Data type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Buffer pool event monitoring

### Data generated by buffer pool event monitors

Buffer pool event monitors produce aggregate metrics about buffer pool activity. You can choose to have the output from a database event monitor to regular tables, files or pipes.

Regardless of the output format you choose, all buffer pool event data comes from the event\_bufferpool logical data group. In addition, if you choose to have statement event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a buffer pool event monitor:

Information written by a buffer pool event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a buffer pool event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

*Table 74. Tables produced by BUFFERPOOL write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name            | Logical data groups reported  |
|-------------------------------|---|
| BUFFERPOOL_ <i>evmon-name</i> | event_bufferpool  |
| CONTROL_ <i>evmon-name</i>    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

### Tables produced

*Table 75. Information returned for a buffer pool event monitor: Default table name: BUFFERPOOL\_*evmon-name**

| Column Name       | Data Type    | Description                               |
|-------------------|--------------|---|
| BLOCK_IOS         | BIGINT       | block_ios - Number of block I/O requests  |
| BP_NAME           | VARCHAR(20)  | bp_name - Buffer pool name                |
| DB_NAME           | CHARACTER(8) | db_name - Database name                   |
| DB_PATH           | VARCHAR(215) | db_path - Database path                   |
| DIRECT_READ_REQS  | BIGINT       | direct_read_reqs - Direct read requests   |
| DIRECT_READ_TIME  | BIGINT       | direct_read_time - Direct read time       |
| DIRECT_READS      | BIGINT       | direct_reads - Direct reads from database |
| DIRECT_WRITE_REQS | BIGINT       | direct_write_reqs - Direct write requests |

Table 75. Information returned for a buffer pool event monitor: Default table name: BUFFERPOOL\_evmon-name (continued)

| Column Name                | Data Type | Description   |
|----------------------------|-----------|---|
| DIRECT_WRITE_TIME          | BIGINT    | direct_write_time - Direct write time                                     |
| DIRECT_WRITES              | BIGINT    | direct_writes - Direct writes to database                                 |
| EVENT_TIME                 | TIMESTAMP | event_time - Event time   |
| EVMON_ACTIVATES            | BIGINT    | evmon_activates - Number of event monitor activations                     |
| EVMON_FLUSHES              | BIGINT    | evmon_flushes - Number of event monitor flushes                           |
| FILES_CLOSED               | BIGINT    | files_closed - Database files closed                                      |
| PAGES_FROM_BLOCK_IOS       | BIGINT    | pages_from_block_ios - Total number of pages read by block I/O            |
| PAGES_FROM_VECTORED_IOS    | BIGINT    | pages_from_vectored_ios - Total pages read by vectored I/O                |
| PARTIAL_RECORD             | SMALLINT  | partial_record - Partial record   |
| POOL_ASYNC_DATA_READ_REQS  | BIGINT    | pool_async_data_read_reqs - Buffer pool asynchronous read requests        |
| POOL_ASYNC_DATA_READS      | BIGINT    | pool_async_data_reads - Buffer pool asynchronous data reads               |
| POOL_ASYNC_DATA_WRITES     | BIGINT    | pool_async_data_writes - Buffer pool asynchronous data writes             |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT    | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_INDEX_READS     | BIGINT    | pool_async_index_reads - Buffer pool asynchronous index reads             |
| POOL_ASYNC_INDEX_WRITES    | BIGINT    | pool_async_index_writes - Buffer pool asynchronous index writes           |
| POOL_ASYNC_READ_TIME       | BIGINT    | pool_async_read_time - Buffer pool asynchronous read time                 |
| POOL_ASYNC_WRITE_TIME      | BIGINT    | pool_async_write_time - Buffer pool asynchronous write time               |
| POOL_ASYNC_XDA_READ_REQS   | BIGINT    | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests     |
| POOL_ASYNC_XDA_READS       | BIGINT    | pool_async_xda_reads - Buffer pool asynchronous XDA data reads            |
| POOL_ASYNC_XDA_WRITES      | BIGINT    | pool_async_xda_writes - Buffer pool asynchronous XDA data writes          |
| POOL_DATA_L_READS          | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                        |
| POOL_DATA_P_READS          | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                       |
| POOL_DATA_WRITES           | BIGINT    | pool_data_writes - Buffer pool data writes                                |
| POOL_INDEX_L_READS         | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                      |
| POOL_INDEX_P_READS         | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                     |
| POOL_INDEX_WRITES          | BIGINT    | pool_index_writes - Buffer pool index writes                              |
| POOL_NO_VICTIM_BUFFER      | BIGINT    | pool_no_victim_buffer - Buffer pool no victim buffers                     |

Table 75. Information returned for a buffer pool event monitor: Default table name: BUFFERPOOL\_evmon-name (continued)

| Column Name           | Data Type | Description   |
|-----------------------|-----------|---|
| POOL_READ_TIME        | BIGINT    | pool_read_time - Total buffer pool physical read time   |
| POOL_WRITE_TIME       | BIGINT    | pool_write_time - Total buffer pool physical write time |
| POOL_XDA_L_READS      | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads   |
| POOL_XDA_P_READS      | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads  |
| POOL_XDA_WRITES       | BIGINT    | pool_xda_writes - Buffer pool XDA data writes           |
| UNREAD_PREFETCH_PAGES | BIGINT    | unread_prefetch_pages - Unread prefetch pages           |
| VECTORED_IOS          | BIGINT    | vectored_ios - Number of vectored I/O requests          |

Table 76. Information returned for a buffer pool event monitor: Default table name: CONTROL\_evmon-name

| Column Name        | Data Type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Table space event monitoring

### Data generated by table space event monitors

Table space event monitors produce aggregate metrics for table spaces in the database. You can choose to have the output from a database event monitor to regular tables, files or pipes.

Regardless of the output format you choose, all table space event data comes from the event\_tablespace logical data group. In addition, if you choose to have statement event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a table space event monitor:

Information written by a table space event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a table space event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

Table 77. Tables produced by TABLESPACE write-to-table event monitors. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by evmon-name in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name    | Logical data groups reported |
|-----------------------|------------------------------|
| TABLESPACE_evmon-name | event_tablespace             |

Table 77. Tables produced by TABLESPACE write-to-table event monitors (continued). The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name         | Logical data groups reported  |
|----------------------------|---|
| CONTROL_ <i>evmon-name</i> | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

### Tables produced

Table 78. Information returned for a table space event monitor: Default table name: TABLESPACE\_*evmon-name*

| Column Name                | Data Type | Description   |
|----------------------------|-----------|---|
| DIRECT_READ_REQS           | BIGINT    | direct_read_reqs - Direct read requests                                   |
| DIRECT_READ_TIME           | BIGINT    | direct_read_time - Direct read time                                       |
| DIRECT_READS               | BIGINT    | direct_reads - Direct reads from database                                 |
| DIRECT_WRITE_REQS          | BIGINT    | direct_write_reqs - Direct write requests                                 |
| DIRECT_WRITE_TIME          | BIGINT    | direct_write_time - Direct write time                                     |
| DIRECT_WRITES              | BIGINT    | direct_writes - Direct writes to database                                 |
| EVENT_TIME                 | TIMESTAMP | event_time - Event time   |
| EVMON_ACTIVATES            | BIGINT    | evmon_activates - Number of event monitor activations                     |
| EVMON_FLUSHES              | BIGINT    | evmon_flushes - Number of event monitor flushes                           |
| FILES_CLOSED               | BIGINT    | files_closed - Database files closed                                      |
| PARTIAL_RECORD             | SMALLINT  | partial_record - Partial record   |
| POOL_ASYNC_DATA_READ_REQS  | BIGINT    | pool_async_data_read_reqs - Buffer pool asynchronous read requests        |
| POOL_ASYNC_DATA_READS      | BIGINT    | pool_async_data_reads - Buffer pool asynchronous data reads               |
| POOL_ASYNC_DATA_WRITES     | BIGINT    | pool_async_data_writes - Buffer pool asynchronous data writes             |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT    | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_INDEX_READS     | BIGINT    | pool_async_index_reads - Buffer pool asynchronous index reads             |
| POOL_ASYNC_INDEX_WRITES    | BIGINT    | pool_async_index_writes - Buffer pool asynchronous index writes           |
| POOL_ASYNC_READ_TIME       | BIGINT    | pool_async_read_time - Buffer pool asynchronous read time                 |
| POOL_ASYNC_WRITE_TIME      | BIGINT    | pool_async_write_time - Buffer pool asynchronous write time               |

Table 78. Information returned for a table space event monitor: Default table name: TABLESPACE\_evmon-name (continued)

| Column Name              | Data Type   | Description   |
|--------------------------|-------------|---|
| POOL_ASYNC_XDA_READ_REQS | BIGINT      | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests |
| POOL_ASYNC_XDA_READS     | BIGINT      | pool_async_xda_reads - Buffer pool asynchronous XDA data reads        |
| POOL_ASYNC_XDA_WRITES    | BIGINT      | pool_async_xda_writes - Buffer pool asynchronous XDA data writes      |
| POOL_DATA_L_READS        | BIGINT      | pool_data_l_reads - Buffer pool data logical reads                    |
| POOL_DATA_P_READS        | BIGINT      | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_DATA_WRITES         | BIGINT      | pool_data_writes - Buffer pool data writes                            |
| POOL_INDEX_L_READS       | BIGINT      | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_INDEX_P_READS       | BIGINT      | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_INDEX_WRITES        | BIGINT      | pool_index_writes - Buffer pool index writes                          |
| POOL_NO_VICTIM_BUFFER    | BIGINT      | pool_no_victim_buffer - Buffer pool no victim buffers                 |
| POOL_READ_TIME           | BIGINT      | pool_read_time - Total buffer pool physical read time                 |
| POOL_WRITE_TIME          | BIGINT      | pool_write_time - Total buffer pool physical write time               |
| POOL_XDA_L_READS         | BIGINT      | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_XDA_P_READS         | BIGINT      | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_XDA_WRITES          | BIGINT      | pool_xda_writes - Buffer pool XDA data writes                         |
| TABLESPACE_FS_CACHING    | SMALLINT    | fs_caching - File system caching                                      |
| TABLESPACE_NAME          | VARCHAR(18) | tablespace_name - Table space name                                    |
| UNREAD_PREFETCH_PAGES    | BIGINT      | unread_prefetch_pages - Unread prefetch pages                         |

Table 79. Information returned for a table space event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Connection event monitoring

### Data generated by connections event monitors

Connections event monitors capture metrics and other monitor elements for each connection to the database by an application. You can choose to have the output written to files, named pipes or regular tables.

Regardless of the output format you choose, all connections event data comes from one of three logical groups:

- event\_connheader
- event\_conn
- event\_connmemuse

In addition, if you choose to have the connections event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a connections event monitor:

Information written by a connections event monitor when the WRITE TO TABLE option is specified.

When you choose WRITE TO TABLE as the output type for the connections event monitor, by default, four tables are produced, each containing monitor elements from one or more logical data groups:

*Table 80. Tables produced by CONNECTIONS write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name            | Logical data groups reported  |
|-------------------------------|---|
| CONNHEADER_ <i>evmon-name</i> | event_connheader  |
| CONN_ <i>evmon-name</i>       | event_conn  |
| CONMEMUSE_ <i>evmon-name</i>  | event_connmemuse  |
| CONTROL_ <i>evmon-name</i>    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

To restrict the output of the event monitor to specific tables, specify the names of the logical groups for which you want tables produced in the CREATE EVENT MONITOR or ALTER EVENT MONITOR statements. Refer to the reference topics for those statements for details.

For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

### Tables produced

*Table 81. Information returned for a connections event monitor: Default table name: CONNHEADER\_*evmon-name**

| Column name     | Data type    | Description  |
|-----------------|--------------|--|
| AGENT_ID        | BIGINT       | agent_id - Application handle (agent ID)                 |
| APPL_ID         | VARCHAR(64)  | appl_id - Application ID                                 |
| APPL_NAME       | VARCHAR(255) | appl_name - Application name                             |
| AUTH_ID         | VARCHAR(128) | auth_id - Authorization ID                               |
| CLIENT_DB_ALIAS | CHAR(8)      | client_db_alias - Database alias used by application     |
| CLIENT_NNAME    | VARCHAR(20)  | “client_nname - Client name monitor element” on page 713 |
| CLIENT_PID      | BIGINT       | client_pid - Client process ID                           |

Table 81. Information returned for a connections event monitor: Default table name: CONNHEADER\_evmon-name (continued)

| Column name     | Data type    | Description                                       |
|-----------------|--------------|---|
| CLIENT_PLATFORM | INTEGER      | client_platform - Client operating platform       |
| CLIENT_PRDID    | VARCHAR(20)  | client_prdid - Client product and version ID      |
| CLIENT_PROTOCOL | INTEGER      | client_protocol - Client communication protocol   |
| CODEPAGE_ID     | INTEGER      | codepage_id - ID of code page used by application |
| CONN_TIME       | TIMESTAMP    | conn_time - Time of database connection           |
| CORR_TOKEN      | VARCHAR(64)  | corr_token - DRDA correlation token               |
| EXECUTION_ID    | VARCHAR(128) | execution_id - User login ID                      |
| SEQUENCE_NO     | CHAR(5)      | sequence_no - Sequence number                     |
| TERRITORY_CODE  | INTEGER      | territory_code - Database territory code          |

Table 82. Information returned for a connections event monitor: Table name: CONN\_evmon-name

| Column Name          | Data Type     | Description                                     |
|----------------------|---------------|---|
| ACC_CURS_BLK         | BIGINT        | acc_curs_blk - Accepted block cursor requests   |
| AGENT_ID             | BIGINT        | agent_id - Application handle (agent ID)        |
| APPL_ID              | VARCHAR(64)   | appl_id - Application ID                        |
| APPL_PRIORITY        | BIGINT        | appl_priority - Application agent priority      |
| APPL_PRIORITY_TYPE   | BIGINT        | appl_priority_type - Application priority type  |
| APPL_SECTION_INSERTS | BIGINT        | appl_section_inserts - Section inserts          |
| APPL_SECTION_LOOKUPS | BIGINT        | appl_section_lookups - Section lookups          |
| APPL_STATUS          | BIGINT        | appl_status - Application status                |
| AUTHORITY_BITMAP     | CHARACTER(22) | authority_bitmap - User authorization level     |
| AUTHORITY_LVL        | BIGINT        | authority_lvl - User authorization level        |
| BINDS_PRECOMPILES    | BIGINT        | binds_precompiles - Binds/precompiles attempted |
| CAT_CACHE_HEAP_FULL  | BIGINT        |   |
| CAT_CACHE_INSERTS    | BIGINT        | cat_cache_inserts - Catalog cache inserts       |
| CAT_CACHE_LOOKUPS    | BIGINT        | cat_cache_lookups - Catalog cache lookups       |
| CAT_CACHE_OVERFLOWES | BIGINT        | cat_cache_overflows - Catalog cache overflows   |

Table 82. Information returned for a connections event monitor: Table name: CONN\_evmon-name (continued)

| Column Name                | Data Type | Description   |
|----------------------------|-----------|---|
| CAT_CACHE_SIZE_TOP         | BIGINT    | cat_cache_size_top - Catalog cache high watermark             |
| COMMIT_SQL_STMTS           | BIGINT    | commit_sql_stmts - Commit statements attempted                |
| COORD_NODE                 | BIGINT    | coord_node - Coordinating node                                |
| DDL_SQL_STMTS              | BIGINT    | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| DEADLOCKS                  | BIGINT    | deadlocks - Deadlocks detected                                |
| DIRECT_READ_REQS           | BIGINT    | direct_read_reqs - Direct read requests                       |
| DIRECT_READ_TIME           | BIGINT    | direct_read_time - Direct read time                           |
| DIRECT_READS               | BIGINT    | direct_reads - Direct reads from database                     |
| DIRECT_WRITE_REQS          | BIGINT    | direct_write_reqs - Direct write requests                     |
| DIRECT_WRITE_TIME          | BIGINT    | direct_write_time - Direct write time                         |
| DIRECT_WRITES              | BIGINT    | direct_writes - Direct writes to database                     |
| DISCONN_TIME               | TIMESTAMP | disconn_time - Database deactivation timestamp                |
| DYNAMIC_SQL_STMTS          | BIGINT    | dynamic_sql_stmts - Dynamic SQL statements attempted          |
| ELAPSED_EXEC_TIME          | BIGINT    | elapsed_exec_time - Statement execution elapsed time          |
| EVMON_FLUSHES              | BIGINT    | evmon_flushes - Number of event monitor flushes               |
| FAILED_SQL_STMTS           | BIGINT    | failed_sql_stmts - Failed statement operations                |
| HASH_JOIN_OVERFLOWES       | BIGINT    | hash_join_overflows - Hash join overflows                     |
| HASH_JOIN_SMALL_OVERFLOWES | BIGINT    | hash_join_small_overflows - Hash join small overflows         |
| INT_AUTO_REBINDS           | BIGINT    | int_auto_rebinds - Internal automatic rebinds                 |
| INT_COMMITS                | BIGINT    | int_commits - Internal commits                                |
| INT_DEADLOCK_ROLLBACKS     | BIGINT    | int_deadlock_rollbackes - Internal rollbackes due to deadlock |
| INT_ROLLBACKS              | BIGINT    | int_rollbackes - Internal rollbackes                          |
| INT_ROWS_DELETED           | BIGINT    | int_rows_deleted - Internal rows deleted                      |
| INT_ROWS_INSERTED          | BIGINT    | int_rows_inserted - Internal rows inserted                    |
| INT_ROWS_UPDATED           | BIGINT    | int_rows_updated - Internal rows updated                      |
| LOCK_ESCALS                | BIGINT    | lock_escals - Number of lock escalations                      |

Table 82. Information returned for a connections event monitor: Table name: CONN\_evmon-name (continued)

| Column Name             | Data Type | Description   |
|-------------------------|-----------|---|
| LOCK_TIMEOUTS           | BIGINT    | lock_timeouts - Number of lock timeouts                               |
| LOCK_WAIT_TIME          | BIGINT    | lock_wait_time - Time waited on locks                                 |
| LOCK_WAITS              | BIGINT    | lock_waits - Lock waits   |
| OLAP_FUNC_OVERFLOWS     | BIGINT    | olap_func_overflows - OLAP function overflows                         |
| PARTIAL_RECORD          | SMALLINT  | partial_record - Partial record                                       |
| PKG_CACHE_INSERTS       | BIGINT    | pkg_cache_inserts - Package cache inserts                             |
| PKG_CACHE_LOOKUPS       | BIGINT    | pkg_cache_lookups - Package cache lookups                             |
| POOL_DATA_L_READS       | BIGINT    | pool_data_l_reads - Buffer pool data logical reads                    |
| POOL_DATA_P_READS       | BIGINT    | pool_data_p_reads - Buffer pool data physical reads                   |
| POOL_DATA_WRITES        | BIGINT    | pool_data_writes - Buffer pool data writes                            |
| POOL_INDEX_L_READS      | BIGINT    | pool_index_l_reads - Buffer pool index logical reads                  |
| POOL_INDEX_P_READS      | BIGINT    | pool_index_p_reads - Buffer pool index physical reads                 |
| POOL_INDEX_WRITES       | BIGINT    | pool_index_writes - Buffer pool index writes                          |
| POOL_READ_TIME          | BIGINT    | pool_read_time - Total buffer pool physical read time                 |
| POOL_TEMP_DATA_L_READS  | BIGINT    | pool_temp_data_l_reads - Buffer pool temporary data logical reads     |
| POOL_TEMP_DATA_P_READS  | BIGINT    | pool_temp_data_p_reads - Buffer pool temporary data physical reads    |
| POOL_TEMP_INDEX_L_READS | BIGINT    | pool_temp_index_l_reads - Buffer pool temporary index logical reads   |
| POOL_TEMP_INDEX_P_READS | BIGINT    | pool_temp_index_p_reads - Buffer pool temporary index physical reads  |
| POOL_TEMP_XDA_L_READS   | BIGINT    | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads  |
| POOL_TEMP_XDA_P_READS   | BIGINT    | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_WRITE_TIME         | BIGINT    | pool_write_time - Total buffer pool physical write time               |
| POOL_XDA_L_READS        | BIGINT    | pool_xda_l_reads - Buffer pool XDA data logical reads                 |
| POOL_XDA_P_READS        | BIGINT    | pool_xda_p_reads - Buffer pool XDA data physical reads                |
| POOL_XDA_WRITES         | BIGINT    | pool_xda_writes - Buffer pool XDA data writes                         |
| PREFETCH_WAIT_TIME      | BIGINT    | prefetch_wait_time - Time waited for prefetch                         |

Table 82. Information returned for a connections event monitor: Table name: CONN\_evmon-name (continued)

| Column Name           | Data Type    | Description  |
|-----------------------|--------------|--|
| REJ_CURS_BLK          | BIGINT       | rej_curs_blk - Rejected block cursor requests                |
| ROLLBACK_SQL_STMTS    | BIGINT       | rollback_sql_stmts - Rollback statements attempted           |
| ROWS_DELETED          | BIGINT       | rows_deleted - Rows deleted                                  |
| ROWS_INSERTED         | BIGINT       | rows_inserted - Rows inserted                                |
| ROWS_READ             | BIGINT       | rows_read - Rows read  |
| ROWS_SELECTED         | BIGINT       | rows_selected - Rows selected                                |
| ROWS_UPDATED          | BIGINT       | rows_updated - Rows updated                                  |
| ROWS_WRITTEN          | BIGINT       | rows_written - Rows written                                  |
| SELECT_SQL_STMTS      | BIGINT       | select_sql_stmts - Select SQL statements executed            |
| SEQUENCE_NO           | CHARACTER(5) | sequence_no - Sequence number                                |
| SORT_OVERFLOWS        | BIGINT       | sort_overflows - Sort overflows                              |
| STATIC_SQL_STMTS      | BIGINT       | static_sql_stmts - Static SQL statements attempted           |
| SYSTEM_CPU_TIME       | BIGINT       | system_cpu_time - System CPU time                            |
| TOTAL_HASH_JOINS      | BIGINT       | total_hash_joins - Total hash joins                          |
| TOTAL_HASH_LOOPS      | BIGINT       | total_hash_loops - Total hash loops                          |
| TOTAL_OLAP_FUNCS      | BIGINT       | total_olap_funcs - Total OLAP functions                      |
| TOTAL_SORT_TIME       | BIGINT       | total_sort_time - Total sort time                            |
| TOTAL_SORTS           | BIGINT       | total_sorts - Total sorts                                    |
| UID_SQL_STMTS         | BIGINT       | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| UNREAD_PREFETCH_PAGES | BIGINT       | unread_prefetch_pages - Unread prefetch pages                |
| USER_CPU_TIME         | BIGINT       | user_cpu_time - User CPU time                                |
| X_LOCK_ESCALS         | BIGINT       | x_lock_escalations - Exclusive lock escalations              |
| XQUERY_STMTS          | BIGINT       | xquery_stmts - XQuery Statements Attempted                   |

Table 83. Information returned for a connections event monitor: Table name: CONMEMUSE\_evmon-name

| Column name   | Data type   | Description                                     |
|---------------|-------------|---|
| APPL_ID       | VARCHAR(64) | appl_id - Application ID                        |
| EVMON_FLUSHES | BIGINT      | evmon_flushes - Number of event monitor flushes |
| POOL_CUR_SIZE | BIGINT      | pool_cur_size - Current size of memory pool     |
| POOL_ID       | BIGINT      | pool_id - Memory pool identifier                |
| POOL_LIST_ID  | BIGINT      |   |
| POOL_MAX_SIZE | BIGINT      |   |

Table 83. Information returned for a connections event monitor: Table name: CONMEMUSE\_evmon-name (continued)

| Column name    | Data type | Description                            |
|----------------|-----------|--|
| POOL_WATERMARK | BIGINT    | pool_watermark - Memory pool watermark |

Table 84. Information returned for a connections event monitor: Default table name: CONTROL\_evmon-name

| Column Name    | Data Type   | Description                                     |
|----------------|-------------|---|
| APPL_ID        | VARCHAR(64) | appl_id - Application ID                        |
| EVMON_FLUSHES  | BIGINT      | evmon_flushes - Number of event monitor flushes |
| POOL_CUR_SIZE  | BIGINT      | pool_cur_size - Current size of memory pool     |
| POOL_ID        | BIGINT      | pool_id - Memory pool identifier                |
| POOL_LIST_ID   | BIGINT      |   |
| POOL_MAX_SIZE  | BIGINT      |   |
| POOL_WATERMARK | BIGINT      | pool_watermark - Memory pool watermark          |

## Transaction event monitoring

### Data generated by transaction event monitors

Transaction event monitors record information about database transactions.

**Note:** This event monitor has been deprecated. Its use is no longer recommended and it might be removed in a future release. Use the unit of work event monitor to monitor units of work instead.

Regardless of the output format you choose, all transaction event data comes from two logical groups:

- “event\_xact logical data group” on page 70
- “event\_connheader logical data group” on page 53

In addition, if you choose to have the transaction event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a transaction event monitor:

Information written by a transaction event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a transaction event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

Table 85. Tables produced by TRANSACTION write-to-table event monitors. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name            | Logical data groups reported  |
|-------------------------------|---|
| XACT_ <i>evmon-name</i>       | “event_xact logical data group” on page 70  |
| CONNHEADER_ <i>evmon-name</i> | “event_connheader logical data group” on page 53  |
| CONTROL_ <i>evmon-name</i>    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

### Tables produced

Table 86. Information returned for a transaction event monitor: Default table name: XACT\_*evmon-name*

| Column name         | Data type    | Description   |
|---------------------|--------------|---|
| AGENT_ID            | BIGINT       | agent_id - Application handle (agent ID)                        |
| APPL_ID             | VARCHAR(64)  | appl_id - Application ID  |
| EVMON_FLUSHES       | BIGINT       | evmon_flushes - Number of event monitor flushes                 |
| LOCK_ESCALS         | BIGINT       | lock_escals - Number of lock escalations                        |
| LOCK_WAIT_TIME      | BIGINT       | lock_wait_time - Time waited on locks                           |
| LOCKS_HELD_TOP      | BIGINT       | locks_held_top - Maximum number of locks held                   |
| PARTIAL_RECORD      | SMALLINT     | partial_record - Partial record                                 |
| PREV_UOW_STOP_TIME  | TIMESTAMP    | prev_uow_stop_time - Previous unit of work completion timestamp |
| ROWS_READ           | BIGINT       | rows_read - Rows read   |
| ROWS_WRITTEN        | BIGINT       | rows_written - Rows written                                     |
| SEQUENCE_NO         | CHARACTER    | sequence_no - Sequence number                                   |
| STOP_TIME           | TIMESTAMP    | stop_time - Event stop time                                     |
| SYSTEM_CPU_TIME     | BIGINT       | system_cpu_time - System CPU time                               |
| TPMON_ACC_STR       | VARCHAR(200) | tpmon_acc_str - TP monitor client accounting string             |
| TPMON_CLIENT_APP    | VARCHAR(255) | tpmon_client_app - TP monitor client application name           |
| TPMON_CLIENT_USERID | VARCHAR(255) | tpmon_client_userid - TP monitor client user ID                 |
| TPMON_CLIENT_WKSTN  | VARCHAR(255) | tpmon_client_wkstn - TP monitor client workstation name         |
| UOW_LOG_SPACE_USED  | BIGINT       | uow_log_space_used - Unit of work log space used                |
| UOW_START_TIME      | TIMESTAMP    | uow_start_time - Unit of work start timestamp                   |
| UOW_STATUS          | BIGINT       | uow_status - Unit of work status                                |

Table 86. Information returned for a transaction event monitor: Default table name: XACT\_evmon-name (continued)

| Column name   | Data type | Description                                |
|---------------|-----------|--|
| USER_CPU_TIME | BIGINT    | user_cpu_time - User CPU time              |
| X_LOCK_ESCALS | BIGINT    | x_lock_escals - Exclusive lock escalations |

Table 87. Information returned for a transaction event monitor: Default table name: CONNHEADER\_evmon-name

| Column Name     | Data Type    | Description  |
|-----------------|--------------|--|
| AGENT_ID        | BIGINT       | agent_id - Application handle (agent ID)                 |
| APPL_ID         | VARCHAR(64)  | appl_id - Application ID                                 |
| APPL_NAME       | VARCHAR(255) | appl_name - Application name                             |
| AUTH_ID         | VARCHAR(128) | auth_id - Authorization ID                               |
| CLIENT_DB_ALIAS | CHARACTER(8) | client_db_alias - Database alias used by application     |
| CLIENT_NNAME    | VARCHAR(20)  | “client_nname - Client name monitor element” on page 713 |
| CLIENT_PID      | BIGINT       | client_pid - Client process ID                           |
| CLIENT_PLATFORM | INTEGER      | client_platform - Client operating platform              |
| CLIENT_PRDID    | VARCHAR(20)  | client_prdid - Client product and version ID             |
| CLIENT_PROTOCOL | INTEGER      | client_protocol - Client communication protocol          |
| CODEPAGE_ID     | INTEGER      | codepage_id - ID of code page used by application        |
| CONN_TIME       | TIMESTAMP    | conn_time - Time of database connection                  |
| CORR_TOKEN      | VARCHAR(64)  | corr_token - DRDA correlation token                      |
| EXECUTION_ID    | VARCHAR(128) | execution_id - User login ID                             |
| SEQUENCE_NO     | CHARACTER(5) | sequence_no - Sequence number                            |
| TERRITORY_CODE  | INTEGER      | territory_code - Database territory code                 |

Table 88. Information returned for a transaction event monitor: Default table name: CONTROL\_evmon-name

| Column name        | Data type    | Description                                    |
|--------------------|--------------|--|
| EVENT_MONITOR_NAME | VARCHAR(128) | event_monitor_name - Event monitor name        |
| MESSAGE            | VARCHAR(128) | message - Control table message                |
| MESSAGE_TIME       | TIMESTAMP    | message_time - Timestamp control table message |

## Deadlock event monitoring

### Data generated by deadlock event monitors

Deadlock event monitors record information about deadlock conditions.

**Note:** This event monitor has been deprecated. It is no longer recommended and might be removed in a future release. Use the locking event monitor to monitor deadlocks instead.

Regardless of the output format you choose, all deadlock event data comes from three logical groups:

- “event\_connheader logical data group” on page 53

- “event\_deadlock logical data group” on page 59
- “event\_dlconn logical data group” on page 60

In addition, if you choose to have the transaction event data written to tables, data from an additional group (CONTROL) is used to generate metadata about the event monitor itself.

### Information written to tables for a deadlock event monitor:

Information written by a deadlock event monitor when the WRITE TO TABLE option is specified.

The sections that follows illustrates the output of a deadlock event monitor when the WRITE TO TABLE option is used on the CREATE EVENT MONITOR statement. For information about the output returned when the event monitor writes to a file or named pipe, see “Event monitor self-describing data stream” on page 108.

*Table 89. Tables produced by DEADLOCK write-to-table event monitors.* The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor (as represented by *evmon-name* in the table names shown in the following table) in the CREATE EVENT MONITOR statement.

| Default table name            | Logical data groups reported  |
|-------------------------------|---|
| CONNHEADER_ <i>evmon-name</i> | “event_connheader logical data group” on page 53  |
| DEADLOCK_ <i>evmon-name</i>   | “event_deadlock logical data group” on page 59  |
| DLCONN_ <i>evmon-name</i>     | “event_dlconn logical data group” on page 60  |
| CONTROL_ <i>evmon-name</i>    | The CONTROL logical group consists of selected elements from one or more of the event_dbheader, event_start and event_overflow logical data groups. |

### Tables produced

*Table 90. Information returned for a deadlock event monitor: Default table name: CONNHEADER\_*evmon-name**

| Column Name     | Data Type    | Description  |
|-----------------|--------------|--|
| AGENT_ID        | BIGINT       | agent_id - Application handle (agent ID) monitor element             |
| APPL_ID         | VARCHAR(64)  | appl_id - Application ID monitor element                             |
| APPL_NAME       | VARCHAR(255) | appl_name - Application name monitor element                         |
| AUTH_ID         | VARCHAR(128) | auth_id - Authorization ID monitor element                           |
| CLIENT_DB_ALIAS | CHARACTER(8) | client_db_alias - Database Alias Used by Application monitor element |
| CLIENT_NNAME    | VARCHAR(20)  | “client_nname - Client name monitor element” on page 713             |
| CLIENT_PID      | BIGINT       | client_pid - Client process ID monitor element                       |
| CLIENT_PLATFORM | INTEGER      | client_platform - Client operating platform monitor element          |
| CLIENT_PRDID    | VARCHAR(20)  | client_prdid - Client product and version ID monitor element         |

Table 90. Information returned for a deadlock event monitor: Default table name: CONNHEADER\_evmon-name (continued)

| Column Name     | Data Type    | Description   |
|-----------------|--------------|---|
| CLIENT_PROTOCOL | INTEGER      | client_protocol - Client communication protocol monitor element   |
| CODEPAGE_ID     | INTEGER      | codepage_id - ID of Code Page Used by Application monitor element |
| CONN_TIME       | TIMESTAMP    | conn_time - Time of database connection monitor element           |
| CORR_TOKEN      | VARCHAR(64)  | corr_token - DRDA Correlation Token monitor element               |
| EXECUTION_ID    | VARCHAR(128) | execution_id - User Login ID monitor element                      |
| SEQUENCE_NO     | CHARACTER(5) | sequence_no - Sequence number monitor element                     |
| TERRITORY_CODE  | INTEGER      | territory_code - Database Territory Code monitor element          |

Table 91. Information returned for a deadlock event monitor: Default table name: DEADLOCK\_evmon-name

| Column Name                | Data Type    | Description  |
|----------------------------|--------------|--|
| DEADLOCK_ID                | BIGINT       | deadlock_id - Deadlock Event Identifier monitor element                          |
| DL_CONNS                   | BIGINT       | dl_conns - Connections involved in deadlock monitor element                      |
| EVMON_ACTIVATES            | BIGINT       | evmon_activates - Number of Event Monitor Activations monitor element            |
| ROLLED_BACK_AGENT_ID       | BIGINT       | rolled_back_agent_id - Rolled Back Agent monitor element                         |
| ROLLED_BACK_APPL_ID        | VARCHAR(64)  | rolled_back_appl_id - Rolled Back Application monitor element                    |
| ROLLED_BACK_PARTICIPANT_NO | SMALLINT     | rolled_back_participant_no - Rolled back application participant monitor element |
| ROLLED_BACK_SEQUENCE_NO    | CHARACTER(5) | rolled_back_sequence_no - Rolled Back Sequence Number monitor element            |
| START_TIME                 | TIMESTAMP    | start_time - Event Start Time monitor element                                    |

Table 92. Information returned for a deadlock event monitor: Default table name: DLCONN\_evmon-name

| Column Name        | Data Type   | Description   |
|--------------------|-------------|---|
| AGENT_ID           | BIGINT      | agent_id - Application handle (agent ID) monitor element              |
| APPL_ID            | VARCHAR(64) | appl_id - Application ID monitor element                              |
| APPL_ID_HOLDING_LK | VARCHAR(64) | appl_id_holding_lk - Application ID Holding Lock monitor element      |
| DATA_PARTITION_ID  | INTEGER     | data_partition_id - Data partition identifier monitor element         |
| DEADLOCK_ID        | BIGINT      | deadlock_id - Deadlock Event Identifier monitor element               |
| EVMON_ACTIVATES    | BIGINT      | evmon_activates - Number of Event Monitor Activations monitor element |
| LOCK_ATTRIBUTES    | BIGINT      | lock_attributes - Lock attributes monitor element                     |

Table 92. Information returned for a deadlock event monitor: Default table name: DLCONN\_evmon-name (continued)

| Column Name               | Data Type     | Description  |
|---------------------------|---------------|--|
| LOCK_COUNT                | BIGINT        | lock_count - Lock count monitor element  |
| LOCK_CURRENT_MODE         | BIGINT        | lock_current_mode - Original lock mode before conversion monitor element                                     |
| LOCK_ESCALATION           | SMALLINT      | lock_escalation - Lock escalation monitor element  |
| LOCK_HOLD_COUNT           | BIGINT        | lock_hold_count - Lock hold count monitor element  |
| LOCK_MODE                 | BIGINT        | lock_mode - Lock mode monitor element  |
| LOCK_MODE_REQUESTED       | BIGINT        | lock_mode_requested - Lock mode requested monitor element  |
| LOCK_NAME                 | CHARACTER(13) | lock_name - Lock name monitor element  |
| LOCK_NODE                 | BIGINT        | lock_node - Lock Node monitor element  |
| LOCK_OBJECT_NAME          | BIGINT        | lock_object_name - Lock Object Name monitor element  |
| LOCK_OBJECT_TYPE          | BIGINT        | lock_object_type - Lock object type waited on monitor element  |
| LOCK_RELEASE_FLAGS        | BIGINT        | lock_release_flags - Lock release flags monitor element  |
| LOCK_WAIT_START_TIME      | TIMESTAMP     | lock_wait_start_time - Lock wait start timestamp monitor element   |
| PARTICIPANT_NO            | SMALLINT      | participant_no - Participant within Deadlock monitor element   |
| PARTICIPANT_NO_HOLDING_LK | SMALLINT      | participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application monitor element |
| SEQUENCE_NO               | CHARACTER(5)  | sequence_no - Sequence number monitor element  |
| SEQUENCE_NO_HOLDING_LK    | CHARACTER(5)  | sequence_no_holding_lk - Sequence Number Holding Lock monitor element  |
| START_TIME                | TIMESTAMP     | start_time - Event Start Time monitor element  |
| TABLE_NAME                | VARCHAR(128)  | table_name - Table name monitor element  |
| TABLE_SCHEMA              | VARCHAR(128)  | table_schema - Table schema name monitor element   |
| TABLESPACE_NAME           | VARCHAR(18)   | tablespace_name - Table space name monitor element   |

Table 93. Information returned for a deadlock event monitor: Default table name: CONTROL\_evmon-name

| Column Name        | Data Type   | Description   |
|--------------------|-------------|---|
| AGENT_ID           | BIGINT      | agent_id - Application handle (agent ID) monitor element              |
| APPL_ID            | VARCHAR(64) | appl_id - Application ID monitor element                              |
| APPL_ID_HOLDING_LK | VARCHAR(64) | appl_id_holding_lk - Application ID Holding Lock monitor element      |
| DATA_PARTITION_ID  | INTEGER     | data_partition_id - Data partition identifier monitor element         |
| DEADLOCK_ID        | BIGINT      | deadlock_id - Deadlock Event Identifier monitor element               |
| EVMON_ACTIVATES    | BIGINT      | evmon_activates - Number of Event Monitor Activations monitor element |

Table 93. Information returned for a deadlock event monitor: Default table name: CONTROL\_evmon-name (continued)

| Column Name               | Data Type     | Description  |
|---------------------------|---------------|--|
| LOCK_ATTRIBUTES           | BIGINT        | lock_attributes - Lock attributes monitor element  |
| LOCK_COUNT                | BIGINT        | lock_count - Lock count monitor element  |
| LOCK_CURRENT_MODE         | BIGINT        | lock_current_mode - Original lock mode before conversion monitor element                                     |
| LOCK_ESCALATION           | SMALLINT      | lock_escalation - Lock escalation monitor element  |
| LOCK_HOLD_COUNT           | BIGINT        | lock_hold_count - Lock hold count monitor element  |
| LOCK_MODE                 | BIGINT        | lock_mode - Lock mode monitor element  |
| LOCK_MODE_REQUESTED       | BIGINT        | lock_mode_requested - Lock mode requested monitor element  |
| LOCK_NAME                 | CHARACTER(13) | lock_name - Lock name monitor element  |
| LOCK_NODE                 | BIGINT        | lock_node - Lock Node monitor element  |
| LOCK_OBJECT_NAME          | BIGINT        | lock_object_name - Lock Object Name monitor element  |
| LOCK_OBJECT_TYPE          | BIGINT        | lock_object_type - Lock object type waited on monitor element  |
| LOCK_RELEASE_FLAGS        | BIGINT        | lock_release_flags - Lock release flags monitor element  |
| LOCK_WAIT_START_TIME      | TIMESTAMP     | lock_wait_start_time - Lock wait start timestamp monitor element   |
| PARTICIPANT_NO            | SMALLINT      | participant_no - Participant within Deadlock monitor element   |
| PARTICIPANT_NO_HOLDING_LK | SMALLINT      | participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application monitor element |
| SEQUENCE_NO               | CHARACTER(5)  | sequence_no - Sequence number monitor element  |
| SEQUENCE_NO_HOLDING_LK    | CHARACTER(5)  | sequence_no_holding_lk - Sequence Number Holding Lock monitor element  |
| START_TIME                | TIMESTAMP     | start_time - Event Start Time monitor element  |
| TABLE_NAME                | VARCHAR(128)  | table_name - Table name monitor element  |
| TABLE_SCHEMA              | VARCHAR(128)  | table_schema - Table schema name monitor element   |
| TABLESPACE_NAME           | VARCHAR(18)   | tablespace_name - Table space name monitor element   |

## Change history event monitoring

The change history event monitor captures information about events on the database server that might impact the running of your regular database workload. You can use the data captured by this event monitor to understand changes in the behavior, performance, or stability of your databases and database management system.

When your regular workload experiences a degradation in performance or unexpected behavior is observed, you can correlate the change in workload behavior with events captured by the change history event monitor. The following changes can have a negative impact on your database system.

- The unexpected creation or dropping of an index

- The failure of scheduled maintenance to run
- The changing of a database configuration parameter or DB2 registry variable

Changes can be explicitly caused by a user. For example, an administrator might run a DDL statement that drops an index. Or, changes might occur implicitly or automatically without any user interaction. For example, the self-tuning memory manager (STMM) might change a configuration parameter, or automatic table reorganization might reorganize a table.

Manually tracking changes to the database server can be a difficult task. Historically, the information for different types of changes has been captured through different interfaces. For example, configuration updates are written to the diagnostic log files (for example, the db2diag log files), while utility progress is captured in the database history file. The change history event monitor provides you with a single interface that captures the events that change the behavior and performance characteristics of your database system. Using the event monitor tables, you can investigate any change events that are of interest.

The change history event monitor can capture change-related events for a number of actions and operations, including:

- Database and database manager configuration parameter changes
- Registry variable changes
- Execution of DDL statements
- Change history event monitor startup
- Execution of the following DB2 utilities and commands:
  - LOAD
  - ADMIN\_MOVE\_TABLE procedure invocations
  - BACKUP DATABASE (ONLINE option only)
  - RESTORE DATABASE (ONLINE option only)
  - ROLLFORWARD DATABASE (ONLINE option only)
  - REDISTRIBUTE DATABASE PARTITION GROUP
  - REORG
  - RUNSTATS

Generally, information related to events that occur while the change history event monitor is inactive or the database is offline are not captured. However, the change history event monitor can be configured to capture the registry variable values that are in effect when the an event monitor is activated. Similarly, database and database manager configuration parameter values can be captured when a change history event monitor is activated. When capturing configuration parameter values, the event monitor can detect if any configuration parameters were changed while the event monitor was inactive, and so the event monitor captures configuration parameter values only if changes occur.

### **Data generated by change history event monitors**

Change history event monitors capture information about activities that might impact the performance, behavior, and stability of databases and database management systems. The output from a change history event monitor is written to logical data groups, where each logical data group has an associated event monitor table.

A change-related action can generate one or more events in the change history event monitor. For example, a database configuration update generates a single event, while the execution of the REORG utility generates two events that mark

the beginning and end of the REORG operation. There is a one-to-many mapping between an event and a logical data group. An event can write information to more than one logical data group and can write more than one entry (row) to the table associated with a given logical data group. Each change-related event is uniquely identified by the following three key fields:

**Event timestamp**

The time that the event occurred.

**Event ID**

A numeric token that ensures uniqueness in cases where the event timestamp is common.

**Member**

The database manager process where the event occurred.

All logical groups contain these three fields and all records or rows corresponding to the same event contain the same values for these fields. These common values facilitate the joining of information across different logical data groups. Utility operations and configuration parameter updates on different members are captured as different events and result in different values for these key fields.

The change history event monitor only supports the TABLE target for event monitor logical data groups. The change history event monitor does not support UNFORMATTED EVENT TABLE, FILE, and PIPE targets.

The following table contains a list of the logical data groups and associated tables used by the change history event monitor. The default table name for each logical data group is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor when it was created using the CREATE EVENT MONITOR statement. The table names shown are the default table names when a name is not specified as part of the CREATE EVENT MONITOR statement.

*Table 94. Logical data groups for the change history event monitor*

| Logical data group | Default table name  | Contains   |
|--------------------|---|--|
| CHANGESUMMARY      | CHANGESUMMARY_evmon-name (see "CHANGESUMMARY logical data group" on page 371) | Summary of all events captured by the change history event monitor |
| DBDBMCFG           | DBDBMCFG_evmon-name (see "DBDBMCFG logical data group" on page 375)           | Configuration parameter changes                                    |
| REGVAR             | REGVAR_evmon-name (see "REGVAR logical data group" on page 377)               | Registry variable changes  |
| DDLSTMTEXEC        | DDLSTMTEXEC_evmon-name (see "DDLSTMTEXEC logical data group" on page 378)     | DDL execution  |
| TXNCOMPLETION      | TXNCOMPLETION_evmon-name (see "TXNCOMPLETION logical data group" on page 381) | Occurrence of a commit, rollback, or rollback to savepoint         |
| EVMONSTART         | EVMONSTART_evmon-name (see "EVMONSTART logical data group" on page 382)       | Event monitor startup information                                  |

Table 94. Logical data groups for the change history event monitor (continued)

| Logical data group | Default table name  | Contains                         |
|--------------------|---|----------------------------------|
| UTILSTART          | UTILSTART_evmon-name (see "UTILSTART logical data group" on page 383)       | Utility startup information      |
| UTILLOCATION       | UTILLOCATION_evmon-name (see "UTILLOCATION logical data group" on page 387) | Utility path or file information |
| UTILSTOP           | UTILSTOP_evmon-name (see "UTILSTOP logical data group" on page 389)         | Utility stop information         |
| UTILPHASE          | UTILPHASE_evmon-name (see "UTILPHASE logical data group" on page 391)       | Utility phase information        |

The change history event monitor can capture a wide range of events. Not all events are of interest to all users. You can control which event types are captured by the change history event monitor by using the WHERE EVENT IN clause in the CREATE EVENT MONITOR statement when you create the event monitor.

The following table shows the actions on the database server that generate events available for capture by the change history event monitor. It also indicates which control options specified in the WHERE EVENT IN clause results in the capture of these events and which logical data groups are populated when the event is generated.

Table 95. Events generated by actions

| Action  | Event type   | WHERE EVENT IN clause         | Logical data group        | Details  |
|---|--------------|-------------------------------|---------------------------|--|
| Changing a database configuration parameter   | DBCFCG       | ALL<br>CFGALL<br>DBCFCG       | CHANGESUMMARY<br>DBDBMCFG | 1 record written to CHANGESUMMARY<br>1 record written to DBDBMCFG for each changed parameter |
| Capturing all database configuration parameter values at event monitor startup if a database configuration parameter was changed while the event monitor was inactive | DBCFCGVALUES | ALL<br>CFGALL<br>DBCFCGVALUES | CHANGESUMMARY<br>DBDBMCFG | 1 record written to CHANGESUMMARY<br>1 record written to DBDBMCFG for each parameter         |
| Changing a database manager configuration parameter   | DBMCFG       | ALL<br>CFGALL<br>DBMCFG       | CHANGESUMMARY<br>DBDBMCFG | 1 record written to CHANGESUMMARY<br>1 record written to DBDBMCFG for each changed parameter |

Table 95. Events generated by actions (continued)

| Action  | Event type    | WHERE EVENT IN clause   | Logical data group             | Details   |
|---|---------------|---|--------------------------------|---|
| Capturing all database manager configuration parameter values at event monitor startup if a database manager configuration parameter was changed while the event monitor was inactive | DBMCFGVALUES  | ALL<br>CFGALL<br>DBMCFGVALUES   | CHANGESUMMARY<br>DBDBMCFG      | 1 record written to CHANGESUMMARY<br>1 record written to DBDBMCFG for each parameter  |
| Changing a registry variable. Only immediate updates (registry variable changes that use the -immediate flag on the <b>db2set</b> command) generate events.                           | REGVAR        | ALL<br>CFGALL<br>REGVAR   | CHANGESUMMARY<br>REGVAR        | 1 record written to CHANGESUMMARY<br>1 record written to REGVAR for each changed variable. Only registry variables that are set explicitly are captured. No records are written for variables that are implicitly set through an aggregate registry variable. |
| Capturing registry variable values at event monitor startup   | REGVARVALUES  | ALL<br>CFGALL<br>REGVARVALUES   | CHANGESUMMARY<br>REGVAR        | 1 record written to CHANGESUMMARY<br>1 record written to REGVAR for each explicitly set variable  |
| Executing a DDL statement successfully  | DDLSTMTEXEC   | ALL<br>DDLALL<br>DDLDATA<br>DDLFEDERATED<br>DDLMONITOR<br>DDLSECURITY<br>DDLSQL<br>DDLSTORAGE<br>DDLWLM<br>DDLXML | CHANGESUMMARY<br>DDLSTMTEXEC   | 1 record written to CHANGESUMMARY<br>1 record written to DDLSTMTEXEC  |
| Commit or rollback of a transaction containing a successfully executed DDL statement or the rollback to a savepoint containing a successfully executed DDL statement                  | TXNCOMPLETION | ALL<br>DDLALL<br>DDLDATA<br>DDLFEDERATED<br>DDLMONITOR<br>DDLSECURITY<br>DDLSQL<br>DDLSTORAGE<br>DDLWLM<br>DDLXML | CHANGESUMMARY<br>TXNCOMPLETION | 1 record written to CHANGESUMMARY<br>1 record written to TXNCOMPLETION  |
| Starting or stopping the event monitor  | EVMONSTART    | not applicable  | CHANGESUMMARY<br>EVMONSTART    | 1 record written to CHANGESUMMARY<br>1 record written to EVMONSTART   |

Table 95. Events generated by actions (continued)

| Action  | Event type     | WHERE EVENT IN clause  | Logical data group                         | Details  |
|---|----------------|--|--|--|
| Starting utility execution or resuming execution after being paused. This event is only generated on the coordinator member | UTILSTART      | ALL<br>BACKUP<br>LOAD<br>MOVETABLE<br>REDISTRIBUTE<br>REORG<br>RESTORE<br>ROLLFORWARD<br>RUNSTATS<br>UTILALL | CHANGESUMMARY<br>UTILSTART<br>UTILLOCATION | 1 record written to CHANGESUMMARY<br>1 record written to UTILSTART<br>0 or more records written to UTILLOCATION; 1 record for each file associated with a utility start. |
| Completing utility execution or pausing execution. This event is only generated on the coordinator member                   | UTILSTOP       | ALL<br>BACKUP<br>LOAD<br>MOVETABLE<br>REDISTRIBUTE<br>REORG<br>RESTORE<br>ROLLFORWARD<br>RUNSTATS<br>UTILALL | CHANGESUMMARY<br>UTILSTOP                  | 1 record written to CHANGESUMMARY<br>1 record written to UTILSTOP  |
| Processing of a utility starts on a member. This event is only generated in a multi-member environment                      | UTILSTARTPROC  | ALL<br>BACKUP<br>LOAD<br>MOVETABLE<br>REDISTRIBUTE<br>REORG<br>RESTORE<br>ROLLFORWARD<br>RUNSTATS<br>UTILALL | CHANGESUMMARY<br>UTILSTART                 | 1 record written to CHANGESUMMARY<br>1 record written to UTILSTART   |
| Processing of a utility stops on a member. This event is only generated in a multi-member environment                       | UTILSTOPPROC   | ALL<br>BACKUP<br>LOAD<br>MOVETABLE<br>REDISTRIBUTE<br>REORG<br>RESTORE<br>ROLLFORWARD<br>RUNSTATS<br>UTILALL | CHANGESUMMARY<br>UTILSTOP                  | 1 record written to CHANGESUMMARY<br>1 record written to UTILSTOP  |
| Starting the execution of a specific processing phase of a utility on a member  | UTILPHASESTART | ALL<br>BACKUP<br>UTILALL   | CHANGESUMMARY<br>UTILPHASE                 | 1 record written to CHANGESUMMARY<br>1 record written to UTILPHASE   |
| Stopping the execution of a specific processing phase of a utility on a member  | UTILPHASESTOP  | ALL<br>BACKUP<br>UTILALL   | CHANGESUMMARY<br>UTILPHASE                 | 1 record written to CHANGESUMMARY<br>1 record written to UTILPHASE   |

**CHANGESUMMARY logical data group:**

The table for the CHANGESUMMARY logical data group is produced by the change history event monitor, where each row represents a unique change history

event that occurred. This table provides you with a quick method of determining whether any changes were captured by the change history event monitor and which other logical data groups contain the details of those changes.

The following table provides a summary of the change event information collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

*Table 96. Information returned by a change history event monitor for the CHANGESUMMARY logical data group. The default table name is CHANGESUMMARY\_evmon-name.*

| Column name     | Data type   | Description   |
|-----------------|-------------|---|
| EVENT_ID        | BIGINT      | A unique token associated with the event.   |
| EVENT_TIMESTAMP | TIMESTAMP   | The time the event was generated.   |
| MEMBER          | SMALLINT    | The member where the event occurred.  |
| EVENT_TYPE      | VARCHAR(32) | The type of event that occurred. For this logical data group, the type is one of: <ul style="list-style-type: none"> <li>• DBCFG</li> <li>• DBCFGVALUES</li> <li>• DBMCFG</li> <li>• DBMCFGVALUES</li> <li>• REGVAR</li> <li>• REGVARVALUES</li> <li>• DDLSTMTEXEC</li> <li>• TXNCOMPLETION</li> <li>• EVMONSTART</li> <li>• UTILSTART</li> <li>• UTILSTOP</li> <li>• UTILSTARTPROC</li> <li>• UTILSTOPPROC</li> <li>• UTILPHASESTART</li> <li>• UTILPHASESTOP</li> </ul> |
| COORD_MEMBER    | SMALLINT    | The coordinating member for the given unit of work or workload.   |

Table 96. Information returned by a change history event monitor for the CHANGESUMMARY logical data group. The default table name is CHANGESUMMARY\_evmon-name. (continued)

| Column name           | Data type                   | Description   |
|-----------------------|-----------------------------|---|
| UTILITY_INVOCATION_ID | VARCHAR(32)<br>FOR BIT DATA | A unique identifier corresponding to the utility invocation that is captured for EVENT_TYPEs:<br><ul style="list-style-type: none"> <li>• UTILSTART</li> <li>• UTILSTOP</li> <li>• UTILSTARTPROC</li> <li>• UTILSTOPPROC</li> <li>• UTILPHASESTART</li> <li>• UTILPHASESTOP</li> </ul> Other EVENT_TYPEs result in an empty string. |
| UTILITY_TYPE          | VARCHAR(16)                 | If the UTILITY_INVOCATION_ID is not empty, then the type is one of:<br><ul style="list-style-type: none"> <li>• BACKUP</li> <li>• LOAD</li> <li>• MOVETABLE</li> <li>• REDISTRIBUTE</li> <li>• REORG</li> <li>• RESTORE</li> <li>• ROLLFORWARD</li> <li>• RUNSTATS</li> </ul> otherwise an empty string.                            |
| APPL_ID               | VARCHAR(64)                 | The identifier generated when the application connects to the database.   |
| APPL_NAME             | VARCHAR(255)                | The name of the application running at the client, as known to the database.  |
| APPLICATION_HANDLE    | BIGINT                      | A system-wide unique ID for the application   |
| SYSTEM_AUTHID         | VARCHAR(128)                | The system authorization ID for the connection. This is a synonym for the system_auth_id monitor element.   |
| SESSION_AUTHID        | VARCHAR(128)                | The current authorization ID for the session being used by the application. This is a synonym for the session_auth_id monitor element.  |
| CLIENT_PLATFORM       | VARCHAR(12)                 | The operating system on which the client application is running.  |

Table 96. Information returned by a change history event monitor for the CHANGESUMMARY logical data group. The default table name is CHANGESUMMARY\_evmon-name. (continued)

| Column name        | Data type    | Description  |
|--------------------|--------------|--|
| CLIENT_PROTOCOL    | VARCHAR(10)  | The communication protocol that the client application is using to communicate with the server.  |
| CLIENT_PORT_NUMBER | INTEGER      | The port number on the client machine that the application is using to communicate with the database server.   |
| CLIENT_PID         | BIGINT       | The process ID of the client application that made the connection to the database.   |
| CLIENT_HOSTNAME    | VARCHAR(255) | The hostname of the machine the client application is connecting from.   |
| CLIENT_WRKSTNNAME  | VARCHAR(255) | The name identifying the client system or workstation.   |
| CLIENT_ACCTNG      | VARCHAR(200) | The data passed to the target database for logging and diagnostic purposes.  |
| CLIENT_USERID      | VARCHAR(255) | The client user ID provided to the server.   |
| CLIENT_APPLNAME    | VARCHAR(255) | The name identifying the server transaction program that is performing the transaction.  |
| BACKUP_TIMESTAMP   | VARCHAR(14)  | <p>If UTILITY_TYPE is BACKUP and EVENT_TYPE is UTILSTART, the BACKUP_TIMESTAMP value is the timestamp of the backup image.</p> <p>If UTILITY_TYPE is RESTORE and EVENT_TYPE is UTILSTOP, the BACKUP_TIMESTAMP value is the timestamp of the backup image.</p> <p>For all other cases, the BACKUP_TIMESTAMP is an empty string.</p> <p>A BACKUP_TIMESTAMP can be correlated with information stored in the database history file (for example, Lookup sequence information) using the SYSIBMADM.DB_HISTORY administration view.</p> |

### DBDBMCFG logical data group:

The table for the DBDBMCFG logical data group is produced by the change history event monitor, where each row represents a configuration parameter that was updated as part of a DBCFG or DBMCFG event, or captured at event monitor startup as part of a DBCFGVALUES or DBMCFGVALUES event. The CFG\_COLLECTION\_TYPE monitor element identifies whether the record describes a configuration parameter update, or an initial value recorded at event monitor startup.

The following table shows the configuration parameter changes collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

*Table 97. Information returned by a change history event monitor for the DBDBMCFG logical data group. The default table name is DBDBMCFG\_evmon-name.*

| Column name     | Data type    | Description  |
|-----------------|--------------|--|
| EVENT_ID        | BIGINT       | A unique token associated with the event.  |
| EVENT_TIMESTAMP | TIMESTAMP    | The time the event was generated.  |
| MEMBER          | SMALLINT     | The member where the event occurred.   |
| EVENT_TYPE      | VARCHAR(32)  | The type of event that occurred. For this logical data group, the type is one of: <ul style="list-style-type: none"><li>• DBCFG</li><li>• DBCFGVALUES</li><li>• DBMCFG</li><li>• DBMCFGVALUES</li></ul>  |
| CFG_NAME        | VARCHAR(128) | The name of the configuration parameter.   |
| CFG_VALUE       | VARCHAR(255) | If the EVENT_TYPE is DBCFG or DBMCFG, this is the new value for the configuration parameter.<br><br>If the EVENT_TYPE is DBCFGVALUES or DBMCFGVALUES, this is the on-disk configuration parameter value. The on-disk configuration parameter value is the most current value and might not be in effect yet. |

Table 97. Information returned by a change history event monitor for the DBDBMCFG logical data group. The default table name is DBDBMCFG\_evmon-name. (continued)

| Column name         | Data type    | Description  |
|---------------------|--------------|--|
| CFG_VALUE_FLAGS     | VARCHAR(32)  | <p>This flag indicates how the new configuration parameter value was determined:</p> <ul style="list-style-type: none"> <li>• AUTOMATIC</li> <li>• COMPUTED</li> <li>• NONE</li> </ul> <p>If the EVENT_TYPE is DBCFGVALUES or DBMCFGVALUES, the flags represent the current on-disk value for the configuration parameter.</p>   |
| CFG_OLD_VALUE       | VARCHAR(255) | <p>If the EVENT_TYPE is DBCFG or DBMCFG, this is the old configuration parameter value.</p> <p>If the EVENT_TYPE is DBCFGVALUES or DBMCFGVALUES, this is the current in-memory configuration parameter value. This is the configuration parameter value currently in use.</p>  |
| CFG_OLD_VALUE_FLAGS | VARCHAR(32)  | <p>This flag indicates how the old configuration parameter value was determined:</p> <ul style="list-style-type: none"> <li>• AUTOMATIC</li> <li>• COMPUTED</li> <li>• NONE</li> </ul> <p>If the EVENT_TYPE is DBCFGVALUES or DBMCFGVALUES, the flags represent the current in-memory value for the configuration parameter.</p> |
| CFG_COLLECTION_TYPE | CHAR(1)      | <p>Indicates when the configuration parameter value was collected:</p> <p><b>I</b>      The initial value that was captured when the event monitor was activated.</p> <p><b>U</b>      Updated value</p>   |

Table 97. Information returned by a change history event monitor for the DBDBMCFG logical data group. The default table name is DBDBMCFG\_evmon-name. (continued)

| Column name | Data type | Description  |
|-------------|-----------|--|
| DEFERRED    | CHAR(1)   | Indicates if a change to a configuration parameter value is deferred:<br><br><b>Y</b> Change deferred until next database activation<br><br><b>N</b> Change takes effect immediately |

### REGVAR logical data group:

The table for the REGVAR logical data group is produced by the change history event monitor, where each row represents a registry variable that was updated as part of a REGVAR event, or captured at event monitor startup as part of a RERVARVALUES event. The REGVAR\_COLLECTION\_TYPE monitor element identifies whether the record describes an immediate registry variable update (*U*) or an initial value recorded at event monitor startup (*I*).

The following table shows registry variable changes collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 98. Information returned by a change history event monitor for the REGVAR logical data group. The default table name is REGVAR\_evmon-name.I

| Column name      | Data type    | Description   |
|------------------|--------------|---|
| EVENT_ID         | BIGINT       | A unique token associated with the event.   |
| EVENT_TIMESTAMP  | TIMESTAMP    | The time the event was generated.   |
| MEMBER           | SMALLINT     | The member where the event occurred.  |
| EVENT_TYPE       | VARCHAR(32)  | The type of event that occurred. For this logical data group, the type is one of:<br><ul style="list-style-type: none"> <li>• REGVAR</li> <li>• REGVARVALUES</li> </ul> |
| REGVAR_NAME      | VARCHAR(256) | The name of the registry variable.  |
| REGVAR_VALUE     | CLOB(2k)     | This is the value for the registry variable. If not set, the value is an empty string.  |
| REGVAR_OLD_VALUE | CLOB(2k)     | This is the old value for the registry variable. If the value was not set, this value is an empty string.   |

Table 98. Information returned by a change history event monitor for the REGVAR logical data group. The default table name is REGVAR\_evmon-name.l (continued)

| Column name            | Data type | Description   |
|------------------------|-----------|---|
| REGVAR_LEVEL           | CHAR(1)   | Indicates the level of the registry variable:<br><b>E</b> Environment<br><b>G</b> Global<br><b>I</b> Instance-level<br><b>P</b> Database partition                        |
| REGVAR_COLLECTION_TYPE | CHAR(1)   | Indicates when the registry variable value was collected:<br><b>I</b> The initial value that was captured when the event monitor was activated.<br><b>U</b> Updated value |

#### DDLSTMTEEXEC logical data group:

The table for the DDLSTMTEEXEC logical data group is produced by the change history event monitor, where each row represents an executed DDL statement event. In a partitioned database environment, rows are captured on the coordinator partition for the DDL execution.

Whenever a row is written to the DDLSTMTEEXEC\_evmon-name table, a row is written to the TXNCOMPLETION\_evmon-name table for each associated transaction state change (commit, rollback, or rollback to savepoint) performed in the same unit of work following the DDL statement. You can then use the GLOBAL\_TRANSACTION\_ID, LOCAL\_TRANSACTION\_ID, and SAVEPOINT\_ID columns in the DDLSTMTEEXEC\_evmon-name table to locate the corresponding state change operation in the TXNCOMPLETION\_evmon-name table and determine whether the DDL was committed or not.

The following table shows the DDL statement execution information collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 99. Information returned by a change history event monitor for the DDLSTMTEEXEC logical data group. The default table name is DDLSTMTEEXEC\_evmon-name.

| Column name     | Data type | Description                               |
|-----------------|-----------|---|
| EVENT_ID        | BIGINT    | A unique token associated with the event. |
| EVENT_TIMESTAMP | TIMESTAMP | The time the event was generated.         |
| MEMBER          | SMALLINT  | The member where the event occurred.      |

Table 99. Information returned by a change history event monitor for the DDLSTMTEEXEC logical data group. The default table name is DDLSTMTEEXEC\_evmon-name. (continued)

| Column name           | Data type   | Description  |
|-----------------------|-------------|--|
| EVENT_TYPE            | VARCHAR(32) | The type of event that occurred. For this logical data group, the type is DDLSTMTEEXEC.  |
| GLOBAL_TRANSACTION_ID | VARCHAR(40) | The global transaction ID in use at the time the event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs. |
| LOCAL_TRANSACTION_ID  | VARCHAR(16) | The local transaction ID in use at the time the event occurred. This is the SQLU_TID structure that is part of the transaction logs.                     |
| SAVEPOINT_ID          | BIGINT      | The name of the savepoint set within a unit of work.   |
| UOW_ID                | INTEGER     | The unique identifier for a unit of work ID within an application handle.  |

Table 99. Information returned by a change history event monitor for the DDLSTMTEEXEC logical data group. The default table name is DDLSTMTEEXEC\_evmon-name. (continued)

| Column name        | Data type   | Description  |
|--------------------|-------------|--|
| DDL_CLASSIFICATION | VARCHAR(30) | <p>The classification of the captured DDL:</p> <p><b>DDLSTORAGE</b><br/>The execution of alter database, buffer pool, partition group, storage group, and table space DDL.</p> <p><b>DDLWLM</b><br/>The execution of histogram, service class, threshold, work action set, work class set, and workload DDL.</p> <p><b>DDLMONITOR</b><br/>The execution of event monitor, and usage list DDL.</p> <p><b>DDLSECURITY</b><br/>The execution of audit policy, grant, mask, permission role, revoke, security label, security label component, security policy, and trusted context DDL.</p> <p><b>DDLSQL</b><br/>The execution of alias, function, method, module, package, procedure, schema, synonym, transform, trigger, type, variable, and view DDL.</p> <p><b>DDLDATA</b><br/>The execution of index, sequence, table, and temporary table DDL.</p> |

Table 99. Information returned by a change history event monitor for the DDLSTMTEEXEC logical data group. The default table name is DDLSTMTEEXEC\_evmon-name. (continued)

| Column name                    | Data type   | Description   |
|--------------------------------|-------------|---|
| DDL_CLASSIFICATION (continued) | VARCHAR(30) | <p><b>DDLXML</b><br/>The execution of XSROBJECT DDL.</p> <p><b>DDLFEDERATED</b><br/>The execution of nickname/server, type/user mapping, and wrapper DDL.</p> |
| STMT_TEXT                      | CLOB(2MB)   | The text of the SQL statement.  |

### TXNCOMPLETION logical data group:

The table for the TXNCOMPLETION logical data group is produced by the change history event monitor, where each row represents a completed transaction event. Rows are written whenever a commit, rollback, or rollback to savepoint occurs in the same unit of work as a corresponding DDLSTMTEEXEC\_evmon-name table event.

Use the information in TXNCOMPLETION\_evmon-name table to determine whether the DDL statements in the transaction were committed. You can then use the GLOBAL\_TRANSACTION\_ID, LOCAL\_TRANSACTION\_ID, and SAVEPOINT\_ID columns in the TXNCOMPLETION\_evmon-name table to find the DDL statements in the DDLSTMTEEXEC\_evmon-name table that were affected by a transaction completion event.

The following table shows the transaction completion information collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 100. Information returned by a change history event monitor for the TXNCOMPLETION logical data group. The default table name is TXNCOMPLETION\_evmon-name.

| Column name     | Data type   | Description  |
|-----------------|-------------|--|
| EVENT_ID        | BIGINT      | A unique token associated with the event.  |
| EVENT_TIMESTAMP | TIMESTAMP   | The time the event was generated.  |
| MEMBER          | SMALLINT    | The member where the event occurred.   |
| EVENT_TYPE      | VARCHAR(32) | The type of event that occurred. For this logical data group, the type is TXNCOMPLETION. |

Table 100. Information returned by a change history event monitor for the TXNCOMPLETION logical data group. The default table name is TXNCOMPLETION\_evmon-name. (continued)

| Column name           | Data type   | Description  |
|-----------------------|-------------|--|
| GLOBAL_TRANSACTION_ID | VARCHAR(40) | The global transaction ID in use at the time the event occurred. This is the data field in the SQLP_GXID structure that is part of the transaction logs. |
| LOCAL_TRANSACTION_ID  | VARCHAR(16) | The local transaction ID in use at the time the event occurred. This is the SQLU_TID structure that is part of the transaction logs.                     |
| SAVEPOINT_ID          | BIGINT      | The ID of the savepoint set within a unit of work.   |
| UOW_ID                | INTEGER     | The unique identifier for a unit of work ID within an application handle.  |
| TXN_COMPLETION_STATUS | CHAR(1)     | Indicates the status of the transaction:<br><b>C</b> Commit<br><b>R</b> Rollback<br><b>S</b> Rollback to savepoint                                       |

#### **EVMONSTART logical data group:**

The table for the EVMONSTART logical data group is produced by the change history event monitor, where each row represents a starting of the change history event monitor. Though event monitor startup is not directly related to system performance, this information provides context for other information captured by the monitor.

Event monitor startup events can help you understand when changes started to take effect. For example, activation timestamps help you track when any deferred database or database manager configuration parameter updates took effect. Knowing when the event monitor was activated also helps you understand the completeness of the information captured in the event monitor tables. Any events that occurred while the event monitor is deactivated, either explicitly or implicitly, are not captured. If the database is not activated, then the event monitor is implicitly inactive.

The following table shows the event monitor startup information collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 101. Information returned by a change history event monitor for the *EVMONSTART* logical data group. The default table name is *EVMONSTART\_evmon-name*.

| Column name     | Data type   | Description   |
|-----------------|-------------|---|
| EVENT_ID        | BIGINT      | A unique token associated with the event.   |
| EVENT_TIMESTAMP | TIMESTAMP   | The time the event was generated.   |
| MEMBER          | SMALLINT    | The member where the event occurred.  |
| EVENT_TYPE      | VARCHAR(32) | The type of event that occurred. For this logical data group, the type is <i>EVMONSTART</i>   |
| DB2START_TIME   | TIMESTAMP   | The database member activation timestamp which can be used to track when deferred database manager configuration parameter updates took effect. |
| DB_CONN_TIME    | TIMESTAMP   | The database activation timestamp which can be used to track when deferred database configuration parameter updates took effect.                |

#### UTILSTART logical data group:

The table for the *UTILSTART* logical data group is produced by the change history event monitor, where each row represents a utility that was started.

The following table shows the utility details collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the *CREATE EVENT MONITOR* statement.

Table 102. Information returned by a change history event monitor for the *UTILSTART* logical data group. The default table name is *UTILSTART\_evmon-name*.

| Column name     | Data type   | Description  |
|-----------------|-------------|--|
| EVENT_ID        | BIGINT      | A unique token associated with the event.  |
| EVENT_TIMESTAMP | TIMESTAMP   | The time the event was generated.  |
| MEMBER          | SMALLINT    | The member where the event occurred.   |
| EVENT_TYPE      | VARCHAR(32) | The type of event that occurred. For this logical data group, the type is one of: <ul style="list-style-type: none"> <li>• <i>UTILSTART</i></li> <li>• <i>UTILSTARTPROC</i></li> </ul> |

Table 102. Information returned by a change history event monitor for the UTILSTART logical data group. The default table name is UTILSTART\_evmon-name. (continued)

| Column name           | Data type                   | Description  |
|-----------------------|-----------------------------|--|
| UTILITY_INVOCATION_ID | VARCHAR(32)<br>FOR BIT DATA | A unique identifier corresponding to the utility invocation.   |
| UTILITY_TYPE          | VARCHAR(16)                 | The utility type is one of: <ul style="list-style-type: none"> <li>• BACKUP</li> <li>• LOAD</li> <li>• MOVETABLE</li> <li>• REDISTRIBUTE</li> <li>• REORG</li> <li>• RESTORE</li> <li>• ROLLFORWARD</li> <li>• RUNSTATS</li> </ul> |

Table 102. Information returned by a change history event monitor for the UTILSTART logical data group. The default table name is UTILSTART\_evmon-name. (continued)

| Column name            | Data type | Description   |
|------------------------|-----------|---|
| UTILITY_OPERATION_TYPE | CHAR(1)   | <p>If UTILITY_TYPE is BACKUP, one of:</p> <p><b>D</b> Delta</p> <p><b>I</b> Incremental</p> <p><b>F</b> Full</p> <p>If UTILITY_TYPE is LOAD, one of:</p> <p><b>I</b> Insert</p> <p><b>R</b> Replace</p> <p><b>S</b> Restart</p> <p><b>T</b> Terminate</p> <p>If UTILITY_TYPE is MOVETABLE, one of:</p> <p><b>A</b> Cancel</p> <p><b>C</b> Copy</p> <p><b>I</b> Init</p> <p><b>L</b> Cleanup</p> <p><b>M</b> Move</p> <p><b>R</b> Replay</p> <p><b>S</b> Swap</p> <p><b>V</b> Verify</p> <p>If UTILITY_TYPE is REDISTRIBUTE, one of:</p> <p><b>A</b> Abort</p> <p><b>C</b> Continue</p> <p><b>D</b> Default</p> <p><b>T</b> Target Map</p> <p>If UTILITY_TYPE is REORG, one of:</p> <p><b>A</b> Reorganize all table indexes</p> <p><b>I</b> Index reorganization</p> <p><b>N</b> Inplace table reorganization</p> <p><b>R</b> Reorganize table reclaim extents</p> <p><b>T</b> Classic table reorganization</p> |

Table 102. Information returned by a change history event monitor for the UTILSTART logical data group. The default table name is UTILSTART\_evmon-name. (continued)

| Column name                           | Data type   | Description  |
|---------------------------------------|-------------|--|
| UTILITY_OPERATION_TYPE<br>(continued) | CHAR(1)     | <p>If UTILITY_TYPE is RESTORE, one of:</p> <p><b>A</b> Incremental automatic</p> <p><b>B</b> Incremental abort</p> <p><b>F</b> Full</p> <p><b>M</b> Incremental manual</p> <p>If UTILITY_TYPE is ROLLFORWARD, one of:</p> <p><b>E</b> End of logs</p> <p><b>P</b> Point in time</p> <p>If UTILITY_TYPE is RUNSTATS, one of:</p> <p><b>A</b> All indexes on a table</p> <p><b>I</b> Index</p> <p><b>T</b> Table</p> |
| UTILITY_INVOKER_TYPE                  | VARCHAR(4)  | <p>Indicates how a utility was invoked. One of:</p> <ul style="list-style-type: none"> <li>• AUTO</li> <li>• USER</li> </ul>   |
| UTILITY_PRIORITY                      | INTEGER     | <p>Specifies the amount of relative importance of a throttled utility with respect to its throttled peers. Priority values range 0 - 100, where 0 implies that a utility is executing unthrottled.</p>   |
| UTILITY_START_TYPE                    | VARCHAR(8)  | <p>Indicates how a utility was started. One of:</p> <ul style="list-style-type: none"> <li>• RESUME</li> <li>• START</li> </ul>  |
| OBJECT_TYPE                           | VARCHAR(16) | <p>Type of object the utility acts on. One of:</p> <ul style="list-style-type: none"> <li>• DATABASE</li> <li>• INDEX</li> <li>• PARTITIONGROUP</li> <li>• TABLE</li> <li>• TABLESPACE</li> </ul> <p>This is a synonym for the objtype monitor element.</p>  |

Table 102. Information returned by a change history event monitor for the UTILSTART logical data group. The default table name is UTILSTART\_evmon-name. (continued)

| Column name    | Data type    | Description  |
|----------------|--------------|--|
| OBJECT_SCHEMA  | VARCHAR(128) | If the OBJECT_TYPE is INDEX or TABLE, then the schema of the index or table, otherwise an empty string.  |
| OBJECT_NAME    | VARCHAR(128) | If the OBJECT_TYPE is INDEX, PARTIONGROUP, or TABLE, then the name of the index, partition group, or table.  |
| NUM_TBSPS      | INTEGER      | If the OBJECT_TYPE is DATABASE or TABLESPACE, then the number of table spaces.   |
| TBSP_NAMES     | CLOB(5M)     | If the OBJECT_TYPE is DATABASE or TABLESPACE, a comma delimited list of table space names that the utility acts on.  |
| UTILITY_DETAIL | CLOB(2M)     | A brief description of the work a utility is performing, including some options specified for the utility. For example, a record for the invocation of REORG includes a partially reconstructed command string including some of the different options used by the utility such as access mode . The format of this field is dependent on the type of utility and might change between releases. |

#### UTILLOCATION logical data group:

The table for the UTILLOCATION logical data group is produced by the change history event monitor, where each row represents each file or path associated with the start of a utility.

The following table shows the utility details collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 103. Information returned by a change history event monitor for the UTILLOCATION logical data group. The default table name is UTILLOCATION\_evmon-name.

| Column name | Data type | Description                               |
|-------------|-----------|---|
| EVENT_ID    | BIGINT    | A unique token associated with the event. |

Table 103. Information returned by a change history event monitor for the UTILLOCATION logical data group. The default table name is UTILLOCATION\_evmon-name. (continued)

| Column name           | Data type                   | Description   |
|-----------------------|-----------------------------|---|
| EVENT_TIMESTAMP       | TIMESTAMP                   | The time the event was generated.   |
| MEMBER                | SMALLINT                    | The member where the event occurred.  |
| EVENT_TYPE            | VARCHAR(32)                 | The type of event that occurred. For this logical data group, the type is UTILSTART.  |
| UTILITY_INVOCATION_ID | VARCHAR(32)<br>FOR BIT DATA | A unique identifier corresponding to the utility invocation.  |
| UTILITY_TYPE          | VARCHAR(16)                 | The utility type is one of: <ul style="list-style-type: none"> <li>• BACKUP</li> <li>• LOAD</li> <li>• RESTORE</li> <li>• ROLLFORWARD</li> </ul>  |
| DEVICE_TYPE           | CHAR(1)                     | Identifier for the device type associated with a UTILSTART event. This field determines the interpretation for the LOCATION field. One of: <ul style="list-style-type: none"> <li><b>A</b> TSM</li> <li><b>C</b> Client</li> <li><b>D</b> Disk</li> <li><b>F</b> Snapshot backup</li> <li><b>L</b> Local</li> <li><b>N</b> Internally generated by DB2</li> <li><b>O</b> Other vendor device support</li> <li><b>P</b> Pipe</li> <li><b>Q</b> Cursor</li> <li><b>R</b> Remove fetch data</li> <li><b>S</b> Server</li> <li><b>T</b> Tape</li> <li><b>U</b> User exit</li> <li><b>X</b> X/Open XBSA interface</li> </ul> |

Table 103. Information returned by a change history event monitor for the UTILLOCATION logical data group. The default table name is UTILLOCATION\_evmon-name. (continued)

| Column name   | Data type     | Description   |
|---------------|---------------|---|
| LOCATION_TYPE | CHAR(1)       | <p>Description of what the location is used for.</p> <p>If UTILITY_TYPE is LOAD, one of:</p> <p><b>C</b> Copy target</p> <p><b>D</b> Input data</p> <p><b>L</b> LOB path</p> <p><b>X</b> XML path</p> <p>If UTILITY_TYPE is BACKUP, one of:</p> <p><b>B</b> Backup target location</p> <p>If UTILITY_TYPE is RESTORE, one of:</p> <p><b>S</b> Restore source location</p> <p>If UTILITY_TYPE is ROLLFORWARD, one of:</p> <p><b>O</b> Alternate overflow log path captured as part of the ROLLFORWARD DATABASE command. Note that if the default overflow log path is used, no location record will be captured.</p> <p>otherwise a blank character.</p> |
| LOCATION      | VARCHAR(1024) | Location associated with the event. Locations depend on the UTILITY_TYPE. For example, load input files or backup target path name.   |

#### UTILSTOP logical data group:

The table for the UTILSTOP logical data group is produced by the change history event monitor, where each row represents a utility that has been stopped.

The following table shows the utility details collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 104. Information returned by a change history event monitor for the UTILSTOP logical data group. The default table name is UTILSTOP\_evmon-name.

| Column name  | Data type                | Description  |
|--|--------------------------|--|
| EVENT_ID   | BIGINT                   | A unique token associated with the event.  |
| EVENT_TIMESTAMP  | TIMESTAMP                | The time the event was generated.  |
| MEMBER   | SMALLINT                 | The member where the event occurred.   |
| EVENT_TYPE   | VARCHAR(32)              | The type of event that occurred. For this logical data group, the type is one of: <ul style="list-style-type: none"> <li>UTILSTOP</li> <li>UTILSTOPPROC</li> </ul>   |
| UTILITY_INVOCATION_ID  | VARCHAR(32) FOR BIT DATA | A unique identifier corresponding to the utility invocation.   |
| UTILITY_TYPE   | VARCHAR(16)              | The utility type is one of: <ul style="list-style-type: none"> <li>BACKUP</li> <li>LOAD</li> <li>MOVETABLE</li> <li>REDISTRIBUTE</li> <li>REORG</li> <li>RESTORE</li> <li>ROLLFORWARD</li> <li>RUNSTATS</li> </ul> |
| UTIL_STOP_TYPE   | VARCHAR(8)               | Indicates how a utility was stopped. One of: <ul style="list-style-type: none"> <li>PAUSE</li> <li>STOP</li> </ul>   |
| START_EVENT_ID   | BIGINT                   | Unique identifier of the corresponding UTILSTART or UTILSTARTPROC event. Use with the START_EVENT_TIMESTAMP and member elements to associate the stop record with the corresponding start record.                  |
| START_EVENT_TIMESTAMP  | TIMESTAMP                | Time of the corresponding UTILSTART or UTILSTARTPROC event. Use with the START_EVENT_ID and member elements to associate the stop record with the corresponding start record.                                      |
| SQLCA (SQL communications area) (see <i>SQL Reference Volume 1</i> ) | VARCHAR(8)               | A string that identifies the beginning of the SQL communications area (SQLCA).   |
| SQLABC   | INTEGER                  | Length of the SQL communications area (SQLCA).   |
| SQLCODE  | INTEGER                  | In the SQLCA structure, the SQL return code of the most recently executed SQL statement.   |

Table 104. Information returned by a change history event monitor for the UTILSTOP logical data group. The default table name is UTILSTOP\_evmon-name. (continued)

| Column name | Data type   | Description  |
|-------------|-------------|--|
| SQLERRMC    | VARCHAR(72) | In the SQLCA structure, one or more tokens, separated by 'X'FF', that are substituted for variables in error messages providing specific information about an error condition.             |
| SQLERRP     | VARCHAR(8)  | In the SQLCA structure, a three-letter identifier indicating the product, followed by five alphanumeric characters indicating the version, release, and modification level of the product. |
| SQLERRD1    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLERRD2    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLERRD3    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLERRD4    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLERRD5    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLERRD6    | INTEGER     | See SQLCA (SQL communications area).   |
| SQLWARN     | VARCHAR(11) | In the SQLCA structure, a set of warning indicators, each containing a blank or W.   |
| SQLSTATE    | VARCHAR(5)  | In the SQLCA structure, a return code that indicates the outcome of the most recently executed SQL statement.  |

**UTILPHASE logical data group:**

The table for the UTILPHASE logical data group is produced by the change history event monitor, where each row contains information about the phase of the utility being started or stopped.

Utility execution is divided into phases or processing stages. Currently the change history event monitor only captures the starting and stopping phases of table space backups.

The following table shows the utility phase details collected by the change history event monitor. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor in the CREATE EVENT MONITOR statement.

Table 105. Information returned by a change history event monitor for the UTILPHASE logical data group. The default table name is UTILPHASE\_evmon-name.

| Column name                 | Data type                | Description  |
|-----------------------------|--------------------------|--|
| EVENT_ID                    | BIGINT                   | A unique token associated with the event.  |
| EVENT_TIMESTAMP             | TIMESTAMP                | The time the event was generated.  |
| MEMBER                      | SMALLINT                 | The member where the event occurred.   |
| EVENT_TYPE                  | VARCHAR(32)              | The type of event that occurred. For this logical data group, the type is one of: <ul style="list-style-type: none"> <li>UTILPHASESTART</li> <li>UTILPHASESTOP</li> </ul>  |
| UTILITY_INVOCATION_ID       | VARCHAR(32) FOR BIT DATA | A unique identifier corresponding to the utility invocation.   |
| UTILITY_TYPE                | VARCHAR(16)              | The type of utility that is being started or stopped. For this logical data group, the type is BACKUP.   |
| UTILITY_PHASE_TYPE          | VARCHAR(16)              | If UTILITY_TYPE is BACKUP, the phase type is: <p><b>BACKUPTS</b><br/>Backup table space</p>  |
| PHASE_START_EVENT_ID        | BIGINT                   | If EVENT_TYPE is UTILPHASESTOP, this is the EVENT_ID of corresponding UTILPHASESTART, otherwise -1. Use with the PHASE_START_EVENT_TIMESTAMP and member elements to associate the phase stop record with the corresponding start record. |
| PHASE_START_EVENT_TIMESTAMP | TIMESTAMP                | If EVENT_TYPE is UTILPHASESTOP, this is the time of the corresponding UTILPHASESTART, otherwise empty. Use with the PHASE_START_EVENT_ID and member elements to associate the phase stop record with the corresponding start record.     |
| OBJECT_TYPE                 | VARCHAR(16)              | Type of object the utility acts on. The type is TABLESPACE.<br><br>This is a synonym for the objtype monitor element.  |
| OBJECT_SCHEMA               | VARCHAR(128)             | Reserved for future use  |
| OBJECT_NAME                 | VARCHAR(128)             | If the OBJECT_TYPE is TABLESPACE, then the name of the table.  |
| UTILITY_PHASE_DETAIL        | CLOB(2M)                 | Reserved for future use  |

## Monitoring utility history using the change history event monitor

The change history event monitor can capture a number of events related to the execution of utilities. These events can be used to monitor the history of utility execution on the database server. This event history is written to logical data groups, where each logical data group has an associated event monitor table.

The execution of a utility can generate one or more events in the change history event monitor. For example, the execution of the REORG utility generates two events that mark the beginning and end of the REORG operation. There is a one-to-many mapping between an event and a logical data group. An event can write information to more than one logical data group and can write more than one entry (row) to the table associated with a given logical data group. Each event corresponding to a particular invocation of a utility is identified by the *utility\_invocation\_id* element. The *utility\_invocation\_id* is a binary token that uniquely identifies a given invocation of a utility. The *utility\_invocation\_id* is the same on each member where the utility is executing. The *utility\_invocation\_id* will retain its uniqueness across database deactivation, reactivation, and member shutdown, allowing quick identification of all event monitor records corresponding to a given invocation of a utility. There is no need to join with other fields or worry about duplicate identifiers.

Using the *utility\_invocation\_id* you can identify all events describing a particular invocation of a utility. For example, when the REORG command is issued on a table, a UTILSTART event is generated when the utility starts execution and a UTILSTOP event is generated when the utility completes execution. Both the UTILSTART and UTILSTOP events will have the same *utility\_invocation\_id*, because they describe the same invocation of the REORG command. The *utility\_invocation\_id* can be used to join these events to compute the elapsed time for the utility.

The change history event monitor can monitor the execution of the following utility types:

- BACKUP
- LOAD
- MOVETABLE
- REDISTRIBUTE
- REORG
- RESTORE
- ROLLFORWARD
- RUNSTATS

The change history event monitor will not capture the execution of an offline backup, restore, or rollforward. Note that utility events are only be captured if the change history event monitor is active during the execution of the utility. If the event monitor is deactivated before the utility executes, no events are captured for the execution of that utility. For example, if the utility needs exclusive access to the table space where the event monitor target tables reside.

The following table lists the change history event monitor logical data groups and associated tables are associated with utility execution events. The table name is derived by concatenating the name of the logical data group used to populate the table with the name given to the event monitor when it was created using the

CREATE EVENT MONITOR statement. The table names shown are the default table names when a name is not specified as part of the CREATE EVENT MONITOR statement.

Table 106. Logical data groups populated during utility executions

| Logical data group | Default table name  | Contains   |
|--------------------|---|--|
| CHANGESUMMARY      | CHANGESUMMARY_evmon-name (see "CHANGESUMMARY logical data group" on page 371) | Summary of all events captured by the change history event monitor |
| UTILSTART          | UTILSTART_evmon-name (see "UTILSTART logical data group" on page 383)         | Utility startup information  |
| UTILLOCATION       | UTILLOCATION_evmon-name (see "UTILLOCATION logical data group" on page 387)   | Utility path or file information                                   |
| UTILSTOP           | UTILSTOP_evmon-name (see "UTILSTOP logical data group" on page 389)           | Utility stop information   |
| UTILPHASE          | UTILPHASE_evmon-name (see "UTILPHASE logical data group" on page 391)         | Utility phase information  |

The WHERE EVENT IN clause of the CREATE EVENT MONITOR (change history) statement controls which utilities are monitored by the change history event monitor. The following list indicates which controls enable the capture of which utilities:

**UTILALL**

Capture execution of the load, move table, online backup, online restore, online rollforward, redistribute, reorg and runstats utilities.

**BACKUP**

Capture execution of the online backup utility.

**LOAD**

Capture execution of the load utility.

**MOVETABLE**

Capture execution of the table move utility (invocations of the ADMIN\_MOVE\_TABLE stored procedure).

**REDISTRIBUTE**

Capture execution of the redistribute partition group utility.

**REORG**

Capture execution of the reorg utility.

**RESTORE**

Capture execution of the online restore utility.

**ROLLFORWARD**

Capture execution of the online rollforward utility.

**RUNSTATS**

Capture execution of the runstats utility.

## Collecting change history event data

You can use the change history event monitor to collect information about activities that might impact the performance, behavior, and stability of your databases and database management systems.

### Before you begin

To create a change history event monitor and collect change history event monitor data, you must have DBADM, or SQLADM authority.

### About this task

The change history event monitor captures changes that might impact the running of your regular database workload. When your regular workload experiences a degradation in performance or you observe unexpected behavior, you must determine what changes occurred that might be causing the problem. Each change-related event is uniquely identified by the following three key fields:

#### Event timestamp

The time that the event occurred.

#### Event ID

A numeric token that ensures uniqueness in cases where the event timestamp is common.

#### Member

The database manager process where the event occurred. Member ensures global uniqueness because event timestamp and event ID are only unique per member.

All logical groups contain these three fields and all records or rows corresponding to the same event contain the same values for these fields. These common values facilitate the joining of information across different logical data groups. Utility operations and configuration parameter updates on different members are captured as different events and result in different values for these key fields.

#### Restrictions

Change history event monitor data can be written only to tables associated with logical data groups. The change history event monitor does not write to unformatted event tables, files, or named pipes.

### Procedure

To collect detailed information about activities that might be impacting database performance, behavior, or stability, perform the following steps:

1. Decide which change history events are of interest. The change history event monitor is capable of capturing events describing the following:
  - Configuration parameter changes
  - Registry variable changes
  - DDL execution
  - Occurrence of a commit, rollback, or rollback to savepoint
  - Event monitor startup information
  - Utility startup information
  - Utility path or file information

- Utility stop information
  - Utility phase information
2. Create a change history event monitor called `whats_changed` by using the `CREATE EVENT MONITOR FOR CHANGE HISTORY` statement. Use the `WHERE EVENT IN` clause to specify the change history events that should be captured. The following example shows how to create a change history event monitor that captures all event types:
 

```
CREATE EVENT MONITOR whats_changed
  FOR CHANGE HISTORY WHERE EVENT IN (ALL)
  WRITE TO TABLE
```
  3. Activate the change history event monitor called `whats_changed` by running the following statement:
 

```
SET EVENT MONITOR whats_changed STATE 1
```

## Results

Whenever a change history event takes place while the change history event monitor is active (for example a database configuration update), information about the event will be captured in the change history event monitor tables. Only events identified in the `WHERE EVENT IN` clause of the event monitor will be captured by the change history event monitor.

## Example

### Example: Investigating an increase in lock escalations using the change history event monitor:

You can use the change history event monitor to detect what changes might have led to a database performance degradation.

### Scenario

In this example, users are reporting a decline in database performance. The database administrator (DBA) notices an abnormally high number of lock escalations occurred in the last 24 hours. The DBA also notices a corresponding increase in application lock wait times over the same period.

The DBA has been monitoring configuration changes, index changes, and `LOAD` operations with a change history event monitor. The event monitor was created with the following statement:

```
CREATE EVENT MONITOR CFGHIST
  FOR CHANGE HISTORY WHERE EVENT IN (DBCFCG, DBMCFG, DBCFGVALUES,
  DBMCFGVALUES,REGVAR,REGVARVALUES, DDLDATA, LOAD)
  WRITE TO TABLE
```

The event monitor was activated with the following statement:

```
SET EVENT MONITOR CFGHIST STATE=1
```

The following table shows some sample event monitor data that the `CFGHIST` change history event monitor might write to the `CHANGESUMMARY_CFGHIST` table. All change history event monitors write to the `CHANGESUMMARY` logical data group. As described in “`CHANGESUMMARY` logical data group”, the `CHANGESUMMARY` logical data group returns a number of event monitor elements that summarize the events captured, only a subset of those elements are shown in the following output. The table name is derived by concatenating the

name of the logical data group used to populate the table (CHANGESUMMARY) with the name given to the event monitor in the CREATE EVENT MONITOR statement (CFGHIST).

```

APPL_ID          APPL_NAME  .... EVENT_ID  EVENT_TIMESTAMP
-----
*LOCAL.tripathy.111028110756 db2bp      .... 1      28/10/2011 07:12:02

EVENT_TYPE MEMBER ....
-----
EVMONSTART 0      ....

```

Since performance was not previously a problem, the DBA suspects that some recent change might be causing the problem and performs the following steps:

1. Check the CHANGESUMMARY logical data group for any changes made in the last 24 hours. For this example, assume that the current time is 31/10/2011 06:00:00.

```

SELECT EVENT_TYPE FROM CHANGESUMMARY_CFGHIST
WHERE EVENT_TIMESTAMP > CURRENT_TIMESTAMP - 24 HOURS

```

The query returns the following result:

```

EVENT_TYPE
-----
DBCFCG
DBCFCG

```

The output indicates that there were two database configuration updates in the last 24 hours.

2. Query the DBDBMCFG logical data group to get the details of the configuration changes.
3. **SELECT** EVENT\_TIMESTAMP, CFG\_NAME, CFG\_VALUE, CFG\_OLD\_VALUE, DB\_DEFERRED **FROM** DBDBMCFG\_CHGHIST

The query returns the following result:

```

EVENT_TIMESTAMP  CFG_NAME  CFG_VALUE  CFG_OLD_VALUE  DB_DEFERRED
-----
30/10/2011 08:41:39 LOCKLIST      1024          2048      N
30/10/2011 08:42:35 LOCKTIMEOUT    0             -1         Y

```

The output indicates that locking changes were made during the time period when performance declined.

The DBA notices that the LOCKTIMEOUT change was deferred and issues a query to check whether the database was activated after this configuration change. This check determines whether the configuration change was picked up by the database. If the change was not picked up, then it is unlikely that it is causing the performance problem. The database activation time is recorded in the EVMONSTART logical data group. All change history event monitors write to the EVMONSTART logical data group by default.

```

SELECT COUNT (*)as POST_CFG_ACTIVATIONS FROM EVMONSTART_CHGHIST
WHERE DB_CONN_TIME > TIMESTAMP(2011-10-30-08:42:35)

```

The query returns a nonzero value.

```

POST_CFG_ACTIVATIONS
-----
1

```

This nonzero value confirms that the database was activated after the LOCKTIMEOUT configuration parameter was changed, meaning that the new value is in effect. The DBA now understands what changed on the system and can try adjusting the lock-related configuration parameters back to their original values to see whether this resolves the issue.

**Note:** If the change history event monitor was inactive when the configuration parameters were changed, then the event monitor would not capture the DBCFG events. Instead the change history event monitor would capture a DBCFGVALUES event when the event monitor was started. In the DBDBMCFG logical data group each row represents a configuration parameter that was updated as part of a DBCFG or DBMCFG event, or captured at event monitor startup as part of a DBCFGVALUES or DBMCFGVALUES event. The CFG\_COLLECTION\_TYPE monitor element identifies whether the record describes a configuration parameter update, or an initial value recorded at event monitor startup. The DBA would need to compare the values captured at the start of the current change history event monitor with the values of the previous capture to look for changed values that might be causing the problem. Examining the diagnostic log would also be helpful.

**Example: Identifying configuration changes and utility executions using the change history event monitor:**

You can use the change history event monitor to determine if there have been any recent configuration changes or the recent execution of utilities.

**Scenario**

In this example, the database administrator (DBA) notices a change in database performance over the last 24 hours. The DBA previously created a change history event monitor called HIST, that he has been using to understand changes in the behavior, performance, or stability of the databases and database management system.

The DBA issues the following query against the CHANGESUMMARY logical data group in order to summarize any change events or utility executions that occurred in the last 24 hours.

```

SELECT EVENT_TIMESTAMP,
       EVENT_TYPE,
       UTILITY_TYPE,
       COORD_MEMBER,
       MEMBER
FROM CHANGESUMMARY_HIST
WHERE EVENT_TIMESTAMP > CURRENT_TIMESTAMP - 24 HOURS
ORDER BY BY_EVENT_TIMESTAMP ASC

```

The query might return output similar to the following:

| EVENT_TIMESTAMP            | EVENT_TYPE     | UTILITY_TYPE | COORD_MEMBER | MEMBER |
|----------------------------|----------------|--------------|--------------|--------|
| 2010-10-31-17.29.04.545210 | DBCFG          |              |              | 0 0    |
| 2010-10-31-18.29.04.545210 | UTILSTART      | LOAD         |              | 0 0    |
| 2010-10-31-18.40.04.545210 | UTILSTARTPROC  | LOAD         |              | 0 0    |
| 2010-10-31-18.50.04.545210 | UTILSTOPPROC   | LOAD         |              | 0 0    |
| 2010-10-31-18.40.04.545210 | UTILSTARTPROC  | LOAD         |              | 0 1    |
| 2010-10-31-18.50.04.545210 | UTILSTOPPROC   | LOAD         |              | 0 1    |
| 2010-10-31-19.29.04.545210 | UTILSTOP       | LOAD         |              | 0 0    |
| 2010-10-31-19.56.04.545210 | UTILSTART      | BACKUP       |              | 0 0    |
| 2010-10-31-20.09.04.545210 | UTILPHASESTART | BACKUP       |              | 0 0    |

|                            |               |        |   |   |
|----------------------------|---------------|--------|---|---|
| 2010-10-31-20.29.04.545210 | UTILPHASESTOP | BACKUP | 0 | 0 |
| 2010-10-31-21.29.04.545210 | UTILSTOP      | BACKUP | 0 | 0 |

9 record(s) selected.

From this output, the DBA determines that there have been configuration changes and utility executions in the last 24 hours. He can now query the other change history event monitor logical data groups and obtain more information about the events returned in the CHANGESUMMARY logical data group. For example, for more information about the UTILSTART events, the DBA can query the UTILSTART logical data group to understand which objects are being acted on by the utilities and what options were used when the utilities were started.

### Example: Listing LOAD operations using the change history event monitor:

You can use the change history event monitor to track all LOAD operations performed on a database.

#### Scenario

In this example, the database administrator (DBA) wants to capture and list the history of all load utility executions on a database. To track LOAD utility events.

1. Create a change history event monitor that tracks LOAD events. For example:

```
CREATE EVENT MONITOR MON_LOAD
FOR CHANGE HISTORY WHERE EVENT IN (LOAD)
WRITE TO TABLE
  CHANGESUMMARY (TABLE UTIL_COMMON),
  UTILSTART (TABLE LOAD_START),
  UTILSTOP (TABLE LOAD_STOP)
  UTILLOCATION (TABLE LOAD_INPUT_FILES)
  UTILPHASE (TABLE LOAD_PHASES);
```

2. Activate the event monitor.

```
SET EVENT MONITOR MON_LOAD STATE=1
```

3. Query the logical data groups for information about LOAD operations executed on the database. For example, the following query lists the start and stop time of all executed load utilities. The query shows only the coordinator start and stop times. It ignores pause and resume records in order to show the full elapsed time of utility execution.

```
SELECT A.APPL_ID,
       A.COORD_MEMBER,
       A.EVENT_TIMESTAMP AS START_TIME,
       B.EVENT_TIMESTAMP AS STOP_TIME,
       A.TABLE_SCHEMA,
       A.TABLE_NAME,
       SQLCODE,
       VARCHAR(A.UTILITY_DETAIL, 200) AS DETAIL
FROM   LOAD_START AS A
       LOAD_STOP AS B
       UTIL_COMMON AS C
WHERE  A.UTILITY_INVOCATION_ID = B.UTILITY_INVOCATION_ID AND
       A.UTILITY_START_TYPE = 'START' AND
       B.UTILITY_STOP_TYPE = 'STOP' AND
       A.MEMBER = B.MEMBER AND
       A.MEMBER = A.COORD_MEMBER
ORDER BY A.EVENT_TIMESTAMP ASC
```

The result of the query shows that two load utilities were executed and provides details on their start and stop times, the target (table name) of the load, and details about the loads that were executed.

| APPL_ID                  | START_TIME                 | STOP_TIME                  |
|--------------------------|----------------------------|----------------------------|
| *LOCAL.test.110131213809 | 2010-10-31-17.29.04.545210 | 2010-10-31-17.29.04.545210 |
| *LOCAL.test.110131213809 | 2010-10-31-17.29.04.545210 | 2010-10-31-17.29.04.545210 |

| TABLES_SCHEMA | TABLE_NAME | SQLCODE | DETAIL        |
|---------------|------------|---------|---------------|
| TEST          | T1         | 0       | LOAD CURSOR.. |
| TEST          | T3         | 0       | LOAD DEL...   |

2 record(s) selected.

### Example: Reporting the history of utility execution using the change history event monitor:

You can use the change history event monitor to track utility operations performed on a database.

#### Scenario

In this example, the database administrator (DBA) wants to capture and list the utility event executions on a database. To report utility events:

1. Create a change history event monitor that tracks utility events. For example:

```
CREATE EVENT MONITOR MON_UTIL
FOR CHANGE HISTORY WHERE EVENT IN (UTILALL)
WRITE TO TABLE
  CHANGESUMMARY (TABLE UTIL_COMMON),
  UTILSTART (TABLE UTIL_START),
  UTILSTOP (TABLE UTIL_STOP)
  UTILLOCATION (TABLE UTIL_LOCATION)
  UTILPHASE (TABLE UTIL_PHASES) AUTOSTART;
```

2. Enable the event monitor.

```
SET EVENT MONITOR MON_UTIL STATE=1
```

3. Query the logical data groups for information about utility operations executed on the database. For example, the following query lists the history of each utility invocation per member.

```
WITH UTIL_HIST(TIMESTAMP, UTIL_TYPE, ACTION, PHASE_TYPE, UTILITY_INVOCATION_ID,
MEMBER, SQLCODE) AS
(SELECT A.EVENT_TIMESTAMP,
A.UTILITY_TYPE,
CAST('START' AS VARCHAR(32)),
CAST(NULL AS VARCHAR(16)),
A.UTILITY_INVOCATION_ID,
A.MEMBER,
CAST(NULL as INTEGER)
FROM UTIL_START AS A
UNION ALL
SELECT A.EVENT_TIMESTAMP,
A.UTILITY_TYPE,
CASE WHEN EVENT_TYPE IN ('UTILPHASESTART') THEN
CAST('PHASE START' AS VARCHAR(32))
ELSE
CAST('PHASE STOP' AS VARCHAR(32))
END CASE,
CAST(UTILITY_PHASE_TYPE AS VARCHAR(16)),
A.UTILITY_INVOCATION_ID,
A.MEMBER,
CAST(NULL as INTEGER)
FROM UTIL_PHASE AS B
UNION ALL
SELECT A.EVENT_TIMESTAMP,
A.UTILITY_TYPE,
```

```

        CAST('STOP' AS VARCHAR(32))
        CAST(NULL AS VARCHAR(16)),
        A.UTILITY_INVOCATION_ID,
        A.MEMBER,
        A.SQLCODE
    FROM UTIL_STOP AS C)
SELECT * FROM UTIL_HIST
ORDER BY UTILITY_INVOCATION_ID, MEMBER, TIMESTAMP ASC

```

The resulting report of utility events can be used to:

- Identify any utilities that are overlapping. For example, a utility starts on a partition before another utility has stopped on that same partition.
- Determine where a utility is spending its time. For example, how much time was spent in each phase. Note: In Version 10.1 this is only available for the table space backup phase of online backups.

| TIMESTAMP                  | UTIL_TYPE | ACTION      | PHASE_TYPE | UTILITY_INVOCATION_ID | MEMBER | SQLCODE |
|----------------------------|-----------|-------------|------------|-----------------------|--------|---------|
| 2010-10-31-17.29.04.545210 | LOAD      | START       | -          | x'18A901F...621'      | 0      | -       |
| 2010-10-31-17.50.04.344230 | LOAD      | STOP        | -          | x'18A901F...621'      | 0      | 0       |
| 2010-10-31-17.29.04.545211 | LOAD      | START       | -          | x'18A901F...633'      | 1      | -       |
| 2010-10-31-17.50.04.344229 | LOAD      | STOP        | -          | x'18A901F...633'      | 1      | 0       |
| 2010-10-31-17.29.04.344210 | BACKUP    | START       | -          | x'18A901F...645'      | 0      | -       |
| 2010-10-31-17.50.04.344211 | BACKUP    | PHASE START | BACKUPTS   | x'18A901F...645'      | 0      | 0       |
| 2010-10-31-17.51.04.545214 | BACKUP    | PHASE STOP  | BACKUPTS   | x'18A901F...645'      | 0      | -       |
| 2010-10-31-17.52.04.344218 | BACKUP    | STOP        | -          | x'18A901F...645'      | 0      | 0       |

8 record(s) selected.

### Example: Listing all committed DDL statements using the change history event monitor:

You can use the change history event monitor to quickly list all committed DDL statements executed to determine if any changes made might be impacting your workload.

### Scenario

In this example, the database administrator (DBA) notices a degradation in the performance of a number of queries has occurred in the last 24 hours. The DBA uses the change history event monitor to quickly examine the DDL executed during that time period, in order to determine if any changes were made that might be having a significant impact on the workload (for example, indexes that were dropped). The DBA previously created a change history event monitor called CHGHIST, that is being used to track DDL statements. The DBA issues the following statement to list all committed DDL statements captured by the change history event monitor in the last 24 hours. The issued statement excludes statements that were rolled back, either via a ROLLBACK statement or ROLLBACK TO SAVEPOINT statement.

Note that the change history event monitor records DDL events when the DDL is executed. Whether the DDL causes any change in the database depends on the DDL being committed.

```

WITH savepoint_rollbacks (global_tran_id, local_tran_id, savepoint_id) AS
(SELECT DISTINCT T.global_transaction_id, T.local_transaction_id, T.savepoint_id
 FROM DDLSTMTEXEC_CHGHIST as D, TXNCOMPLETION_CHGHIST as T
 WHERE T.txn_completion_status='S' AND
        D.savepoint_id >= T.savepoint_id AND
        D.event_timestamp <= T.event_timestamp)
SELECT VARCHAR(D.STMT_TEXT, 70) AS STMT_TEXT FROM DDLSTMTEXEC_CHGHIST as D,
        TXNCOMPLETION_CHGHIST as T
 WHERE D.global_transaction_id = T.global_transaction_id AND
        D.local_transaction_id = T.local_transaction_id AND
        T.txn_completion_status = 'C' AND

```

```

(D.global_transaction_id, D.local_transaction_id, D.savepoint_id)
NOT IN (SELECT * FROM savepoint_rollback) AND
D.EVENT_TIMESTAMP > CURRENT_TIMESTAMP - 24 HOURS;

```

STMT\_TEXT

-----  
CREATE INDEX I1 ON T1 (ONE)

1 record(s) selected.

**Example: Listing changes performed by the STMM using the change history event monitor:**

You can use the change history event monitor to list changes performed by the self tuning memory manager (STMM).

**Scenario**

In this example, the database administrator (DBA) wants to monitor any changes performed by the STMM. Since the STMM can modify configuration parameters and buffer pool sizes, the DBA has created an event monitor called HIST to capture configuration and DDL changes.

Changes initiated by the STMM can be found by querying the event monitor for records containing one of the following information:

- The application name (appl\_name) is db2stmm.
- The DDL statement text (stmt\_text) contains a comment with the keyword db2stmm. Note that some DDL changes are performed on behalf of the STMM by other applications.

```

SELECT A.EVENT_TIMESTAMP,
       VARCHAR(A.EVENT_TYPE, 20) AS EVENT_TYPE, A.MEMBER
FROM CHANGESUMMARY_HIST A LEFT OUTER JOIN
DDLSTMTEXC_HIST B
ON A.EVENT_TIMESTAMP = B.EVENT_TIMESTAMP AND
A.MEMBER = B.MEMBER AND
A.EVENT_ID = B.EVENT_ID
WHERE (A.APPL_NAME = 'db2stmm' OR
B.STMT_TEXT LIKE '%db2stmm%');

```

The query might return output similar to the following example:

| EVENT_TIMESTAMP            | EVENT_TYPE | MEMBER |
|----------------------------|------------|--------|
| 2011-04-22-12.12.17.832316 | DBCFCG     | 0      |
| 2011-04-22-12.22.35.227550 | DBCFCG     | 0      |
| 2011-04-22-12.12.17.530274 | DBCFCG     | 0      |
| 2011-04-22-12.12.17.721403 | DBCFCG     | 0      |
| 2011-04-22-12.12.17.776889 | DBCFCG     | 0      |
| 2011-04-22-12.22.35.172119 | DBCFCG     | 0      |
| 2011-04-22-12.12.17.665098 | DBCFCG     | 0      |
| 2011-04-22-12.22.35.116343 | DBCFCG     | 0      |
| 2011-04-22-12.29.47.092822 | DBCFCG     | 0      |
| 2011-04-22-12.29.47.037709 | DBCFCG     | 0      |
| 2011-04-22-12.12.17.600511 | DBCFCG     | 0      |
| 2011-04-22-12.22.35.283320 | DBCFCG     | 0      |
| 2011-04-22-12.29.46.752477 | DBCFCG     | 0      |
| 2011-04-22-12.29.47.148562 | DBCFCG     | 0      |

14 record(s) selected.

---

## Event monitor data retention from release to release

Beginning in DB2 Version 10.1, you can upgrade event monitor output tables after you upgrade the DB2 product. This capability lets you retain any data that might exist in event monitor tables that you had before you upgraded.

As event monitors are enhanced in the DB2 product, the tables they produce might change. For example, new columns might be added to a table for reporting new monitor elements. Before Version 10.1, if you had existing event monitors that wrote to tables that contained data that you wanted to retain, and you wanted to collect the data in the newly-added columns, you were required to manually alter them after upgrading to the new release. This alteration involved adding any of the new columns that you might want to use. If you did not add the new columns, the event monitor would work as it had in the previous release, capturing only the data supported by that the event monitor in that release.

Unformatted event tables that had changed could not be upgraded at all; you were required to drop them and then re-create them.

The `EVMON_UPGRADE_TABLES` stored procedure upgrades the definitions of existing event monitor tables to match those produced by the current level of the DB2 product. This feature lets you keep any existing tables that you might have, along with all the data they contain, eliminating the need to manually alter, or to drop and re-create tables.

**Note:** Starting in Version 10.1, you can also use the `ALTER EVENT MONITOR` statement to add new logical groups to an event monitor. You can use this approach as an alternative to `EVMON_UPGRADE_TABLES` to add logical data groups added in a new release. However, you cannot use `ALTER EVENT MONITOR` to modify logical groups that are already associated with the event monitor; if a logical data group already associated with the event monitor has changed, the only way to modify the event monitor is using the `EVMON_UPGRADE_TABLES` procedure.

The `EVMON_UPGRADE_TABLES` procedure works with both regular and UE tables. For regular tables, the procedure adds any new columns needed, drops old columns that are no longer required, and alters any columns as needed. For UE tables, the procedure adds new columns and modifies existing columns as needed to allow the UE table to be processed by the `db2evmonfmt` tool, or the `EVMON_FORMAT_UE_TO_TABLES` or `EVMON_FORMAT_UE_TO_XML` routines.

**Important:** Any active event monitors must be deactivated for the upgrade process to work properly. The `EVMON_UPGRADE_TABLES` procedure automatically deactivates any active event monitors before it begins upgrading tables. Do not reactivate any event monitors with tables being processed by `EVMON_UPGRADE_TABLES`, or the upgrade process will fail. Any event monitors that were active before the upgrade are activated again after the upgrade has completed.

### Implications of not upgrading event monitor tables

As in past releases, you can choose to not upgrade your event monitor tables. However, any new columns that have been added to the event monitor in the new release will not be populated with data, and will not be available for queries. Also, the values for any monitor elements that previously existed in the old release and that increased in size in the new release might be truncated. For example, if a

monitor element increases in size from VARCHAR(20) to VARCHAR(128) in the new release, and you do not upgrade the previously-existing tables, the column that contains the monitor element values will continue to store only 20 characters of data, even though the system may be sending 128-bytes of data for that monitor element to the event monitor.

### **Upgrading tables produced by EVMON\_FORMAT\_UE\_TO\_TABLES**

When used with UE tables, the EVMON\_UPGRADE\_TABLES procedure upgrades the UE table itself; it has no effect on any regular tables that you might have created using the EVMON\_FORMAT\_UE\_TO\_TABLES procedure. After you use EVMON\_UPGRADE\_TABLES to upgrade a UE table, you can also upgrade the output tables produced by EVMON\_FORMAT\_UE\_TO\_TABLES. As of DB2 Version 10.1 EVMON\_FORMAT\_UE\_TO\_TABLES procedure supports a new option: UPGRADE\_TABLES. When you run the EVMON\_FORMAT\_UE\_TO\_TABLES procedure with this option, any existing tables produced by the procedure are altered so that the table columns match the output produced by the new version of EVMON\_FORMAT\_UE\_TO\_TABLES procedure.

For more information, refer to the reference information for EVMON\_FORMAT\_UE\_TO\_TABLES.

---

## Chapter 4. Other monitoring interfaces

---

### Reports generated using the MONREPORT module

The MONREPORT module generates text reports of monitoring data that you can use to troubleshoot SQL performance problems.

You can generate the following reports using the MONREPORT module:

*Table 107. List of reports generated using the MONREPORT module*

| Report Name                 | Procedure to create report | Main data source / table functions   |
|-----------------------------|----------------------------|--|
| Summary report              | MONREPORT.DBSUMMARY        | MON_GET_SERVICE_SUBCLASS and selected details from MON_GET_CONNECTION and MON_GET_WORKLOAD   |
| Connection report           | MONREPORT.CONNECTION       | MON_GET_CONNECTION   |
| Current Applications report | MONREPORT.CURRENTAPPS      | Includes fields from MON_GET_CONNECTION, MON_GET_UNIT_OF_WORK, WLM_GET_SERVICE_CLASS_AGENTS, WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES  |
| Current SQL report          | MONREPORT.CURRENTSQL       | MON_GET_PKG_CACHE_STMT (For the <b>executable_id</b> obtained from the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function.)   |
| Package Cache report        | MONREPORT.PKGCACHE         | MON_GET_PKG_CACHE_STMT   |
| Current Lock Wait report    | MONREPORT.LOCKWAIT         | Most data from MON_GET_APPL_LOCKWAIT; additional data from MON_GET_CONNECTION, WLM_GET_SERVICE_CLASS_AGENTS, WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES, MON_GET_PKG_CACHE_STMT, MON_GET_TABLE |

Most reports start with a summary section that provides one line of key information for each item in the report. For example, the Connection report contains a one-line summary of each connection. The main body of the report consists of a detailed section for each item in the summary.

Each metric in the report is labeled with the underlying monitor element name (for example: CLIENT\_IDLE\_WAIT\_TIME = 44). To determine what the metric represents, search the Information Center for the monitor element name.

You can customize the reports generated by the MONREPORT module. The MONREPORT module is implemented entirely using SQL and you can obtain the module code from the database catalog and create a customized version.

### Reports for initial diagnosis

An important use of these reports is to troubleshoot SQL performance slowdowns. Each report is designed to answer certain diagnosis questions. Some reports support initial diagnosis, while others support subsequent detailed diagnosis of particular types of problems.

Initial diagnosis involves:

- Determining the problem category, by narrowing the problem down to the aspect or stage of processing that has slowed down.
- Identifying the SQL statements involved in the problem and collecting information about the SQL statements for further analysis.

Table 108. MONREPORT module reports suitable for initial diagnosis

| Procedure name  | Information provided and usage  |
|---|---|
| MONREPORT.DBSUMMARY<br>Part 1: System Performance       | <p>Part 1 of the Summary report provides monitor data for most aspects of processing aggregated across the entire database.</p> <p>This information is useful for answering questions about the aspect or stage of processing that has slowed down. For example:</p> <ul style="list-style-type: none"> <li>• Is the problem inside or outside the data server?</li> <li>• Is there a computing resource bottleneck?</li> <li>• Are requests in a wait state? If so, for what resource?</li> <li>• Is the slowdown located in a particular data server processing component?</li> </ul> |
| MONREPORT.DBSUMMARY<br>Part 2: Application Performance  | <p>Part 2 of the Summary report provides key performance indicators for each connection, workload, and service class.</p> <p>This information is useful for answering questions about the scope of application requests involved in the slowdown. For example:</p> <ul style="list-style-type: none"> <li>• Is this slowdown a general system slowdown that affects much or all the workload?</li> <li>• Is this slowdown limited to SQL statements issued from a particular source such as particular connections, DB2 workloads or DB2 service classes?</li> </ul>                    |
| MONREPORT.DBSUMMARY<br>Part 3: Member level information | <p>Part 3 of the Summary report provides key performance indicators for each member.</p> <p>This information is useful for determining whether the slowdown is isolated to one or a few members.</p>  |
| MONREPORT.CURRENTSQL                                    | <p>The current SQL report provides information about statements that are currently running, in the form of several lists of the top <i>N</i> activities. The statements are ranked by different metrics: processing resource, rows processed, direct reads and direct writes.</p> <p>This information is useful for determining whether the slowdown is isolated to one or a few SQL statements. If the slowdown is isolated to one or a few SQL statements, those statements are likely to appear in this report of top statements.</p>  |

Table 108. MONREPORT module reports suitable for initial diagnosis (continued)

| Procedure name        | Information provided and usage  |
|-----------------------|---|
| MONREPORT.PKGCACHE    | <p>The package cache report provides information about statements that have run recently and are stored in the package cache. This report shows several summaries, each listing the top <i>N</i> activities. The activities are ranked by the following monitor elements:</p> <ul style="list-style-type: none"> <li>• CPU</li> <li>• wait time</li> <li>• rows processed</li> <li>• num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics monitor element (if no member was specified) or num_exec_with_metrics - Number of executions with metrics collected monitor element (if a member was specified)</li> <li>• I/O wait time</li> </ul> <p>This report contains a summary for each of these metrics as well as a report for each execution.</p> <p>This information is useful for determining whether the slowdown is isolated to one or a few SQL statements. If so, those statements are likely to appear at the top in this report. The information per execution can help identify the most costly statements while the information summed across executions can help identify statements with the most impact on the system cumulatively considering both the statement cost and frequency of execution.</p> |
| MONREPORT.CURRENTAPPS | <p>The current applications report show the current processing state for units of work, agents, and activities. The report starts with a summary section showing the number of current connections and activities, as well as a series of summaries, such as the summary of current units of work by workload occurrence state. The body of the report consists of one section for each connection that provides the details of the connection.</p> <p>This information is useful for viewing all the work currently running on the system. This allows you to check for patterns that might identify the problem category.</p>   |

## Reports for detailed diagnosis

After completing the initial diagnosis, you might need to pursue a specialized or detailed set of troubleshooting analyses for the problem category you identified during the initial diagnosis phase.

Table 109. MONREPORT module reports suitable for detailed diagnosis

| Procedure name       | Information provided and usage  |
|----------------------|---|
| MONREPORT.CONNECTION | <p>If the MONREPORT.DBSUMMARY report showed that the slowdown is limited to SQL statements issued from a particular connection, then you can view detailed information about the affected connection.</p> <p>This report contains the same metrics as Part 1 of the MONREPORT.DBSUMMARY report, but it presents this information for each connection.</p> |
| MONREPORT.LOCKWAIT   | <p>If the reports viewed during the initial diagnosis suggest there is a lock wait problem, then you can view detailed information about each lock wait currently in progress.</p> <p>This information includes lock holder and lock requester details, as well as characteristics of the lock held and the lock requested.</p>                           |

## Customizing the MONREPORT module reports

You can customize the existing reports generated by the MONREPORT module or create new reports based on an existing report.

### About this task

You can change the wording or organization of a report; or add, remove, or modify monitor elements included in a report. In addition, you can create a new report based on an existing report.

The MONREPORT module is implemented using SQL statements, including stored procedures and data types. To create a customized version of the MONREPORT module, obtain the module code from the database catalog, then modify and deploy it. The MONREPORT module is available in the SYSIBMADM schema.

Each procedure in the MONREPORT module is used to generate one report. For example, the CONNECTION procedure is used to generate the Connection report. When you invoke a procedure, such as the CONNECTION procedure, to generate a report, the procedure invokes other internal routines to perform the final assembly and formatting of the report. The internal routines perform the following functions:

- Call the DB2 monitoring table functions to obtain monitoring data. For many reports, the routines will call the table functions, wait an interval, then call the table functions again.
- Generate delta values, to record the difference between monitor values at the beginning and end of the monitoring interval.
- Perform calculations to obtain percentages, ratios, sums and aggregations.

You can customize the following internal routines:

Table 110. Routines you can customize

| Name             | Description  | Used by report  |
|------------------|--|---|
| CONNDELTA        | This stored procedure returns a result set of delta values from connection metrics.  | This routine is used in conjunction with the Connection report. |
| COMMONREQMETRICS | This store procedure calculates metrics and formats the report output for metrics that are common to Connection and Summary reports. | MONREPORT.CONNECTION and MONREPORT.DBSUMMARY                    |

The MONREPORT module contains other objects that you don't need to modify to customize this module.

Table 111. Routines that do not need to be customized

| Name             | Description  | Used by report             |
|------------------|--|----------------------------|
| INITMSGCACHE     | This stored procedure retrieves translatable strings that appear in the reports.   | Used in all reports.       |
| SAVE_EXEC_INFO   | A stored procedure used internally to store information about SQL statement sections that were executed.                               | Used by MONREPORT.PKGCACHE |
| db2monreport.src | This file is in binary format and is accessed by the INITMSGCACHE routine. This file contains text strings that appear in the reports. | Used in all reports.       |

Table 112. Data types used in the routines, which cannot be customized

| Name  | Description   | Used by report               |
|---|---|------------------------------|
| Data types with names such as MONMETRICS_CHAR255_TYPE and MONMETRICS_CHAR32_TYPE. | These array data types are used to store monitor elements returned by table functions.                                  | Used in most or all reports. |
| REPORT_TYPE   | An array data type used to store the text output that is returned and displayed when the routine terminates.            | Used in most or all reports. |
| MONEXEC_TYPE  | An array data type used to store the executable ID that uniquely identifies an SQL statement section that was executed. | Used in most or all reports. |

## Procedure

1. Obtain the code.
2. Customize the code to meet your needs. Rename the module to distinguish your custom version from the original and modify the code using any suitable SQL editor or stored procedure builder.

- To add text to the report, make the additions directly to the report procedure. Do not manage text strings using the INITMSGCACHE routine.
- To remove report text obtained using the INITMSGCACHE routine, remove the associated code that uses the cached strings.

For example, to change the MONREPORT.CONNECTION report:

- To add or remove metrics, modify the SQL queries in the CONNDELTA procedure. To add a metric, specify an additional column to be selected.
  - To change the details section of the connection report output, update the COMMONREQMETRICS routine to add calculations and ensure the new metrics show up in the output.
  - To change the summary section of the connection report, update the CONNECTION routine to ensure the new metrics show up in the output.
3. Deploy the customized version of the MONREPORT module. Write a set of CREATE or ALTER statements for the module and its routines. If you are using a graphical SQL editor, the editor will automate some of the deployment steps.

---

## Snapshot monitor

You can use the snapshot monitor to capture information about the database and any connected applications at a specific time. Snapshots are useful for determining the status of a database system.

Taken at regular intervals, they are also useful for observing trends and foreseeing potential problems. Some of the data from the snapshot monitor is obtained from the system monitor. The data available from the system monitor is determined by system monitor switches.

The system monitor accumulates information for a database only while it is active. If all applications disconnect from a database and the database deactivates, then the system monitor data for that database is no longer available. You can keep the database active until your final snapshot has been taken, either by starting the database with the ACTIVATE DATABASE command, or by maintaining a permanent connection to the database.

Snapshot monitoring requires an instance attachment. If there is not an attachment to an instance, then a default instance attachment is created. An instance attachment is usually done implicitly to the instance specified by the DB2INSTANCE environment variable when the first database system monitor API is invoked by the application. It can also be done explicitly, using the ATTACH TO command. Once an application is attached, all system monitor requests that it invokes are directed to that instance. This allows a client to monitor a remote server by simply attaching to the instance on it.

In partitioned database environments, snapshots can be taken at any partition of the instance, or globally using a single instance connection. A global snapshot aggregates the data collected at each partition and returns a single set of values.

In DB2 pureScale environments, snapshots can be taken at any member or globally. A global snapshot aggregates the data collected at each member and returns a single set of values.

You can capture a snapshot from the CLP, from SQL table functions, or by using the snapshot monitor APIs in a C or C++ application. A number of different snapshot request types are available, each returning a specific type of monitoring

data. For example, you can capture a snapshot that returns only buffer pool information, or a snapshot that returns database manager information. Before capturing a snapshot, consider if you need information from monitor elements that are under monitor switch control. If a particular monitor switch is off, the monitor elements under its control will not be collected.

## Access to system monitor data: SYSMON authority

Users that are part of the SYSMON database manager level group have the authority to gain access to database system monitor data. System monitor data is accessed using the snapshot monitor APIs, CLP commands, or SQL table functions.

The SYSMON authority group provides the means to enable users without system administration or system control authorities to access database system monitor data.

Aside from SYSMON authority, the only way to access system monitor data using the snapshot monitor is with system administration or system control authority.

Any user that is part of the SYSMON group or has system administration or system control authority can perform the following snapshot monitor functions:

- CLP Commands:
  - GET DATABASE MANAGER MONITOR SWITCHES
  - GET MONITOR SWITCHES
  - GET SNAPSHOT
  - LIST ACTIVE DATABASES
  - LIST APPLICATIONS
  - LIST DCS APPLICATIONS
  - LIST UTILITIES
  - RESET MONITOR
  - UPDATE MONITOR SWITCHES
- APIs:
  - db2GetSnapshot - Get Snapshot
  - db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot() Output Buffer
  - db2MonitorSwitches - Get/Update Monitor Switches
  - db2ResetMonitor - Reset Monitor
- Snapshot SQL table functions without previously running SYSPROC.SNAP\_WRITE\_FILE

## Capturing database system snapshots by using snapshot administrative views and table functions

Authorized users can capture snapshots of monitor information for a DB2 instance by using snapshot administrative views or snapshot table functions. The snapshot administrative views provide a simple means of accessing data for all database partitions of the connected database. The snapshot table functions allow you to request data for a specific database partition, globally aggregated data, or data from all database partitions. Some snapshot table functions allow you to request data from all active databases.

## Before you begin

You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to capture a database snapshot. To obtain a snapshot of a remote instance, you must first connect to a local database belonging to that instance.

## About this task

While new snapshot table functions might be required in future releases if new monitor data is available, the set of snapshot administrative views will remain the same with new columns added to the view, making the administrative views a good choice for application maintenance over time.

Each snapshot view returns a table with one row per monitored object per database partition with each column representing a monitor element. Each table function returns a table with one row per monitored object for the specified partition. The column names of the returned table correlate with the monitor element names.

For example, a snapshot of general application information for the SAMPLE database is captured as follows by using the SNAPAPPL administrative view:

```
SELECT * FROM SYSIBMADM.SNAPAPPL
```

You can also select individual monitor elements from the returned table. For example, the following statement returns only the **agent\_id** and **appl\_id** monitor elements:

```
SELECT agent_id, appl_id FROM SYSIBMADM.SNAPAPPL
```

### Restrictions

Snapshot administrative views and table functions cannot be used with either of the following:

- Monitor switches commands or APIs
- Monitor reset commands or APIs

This restriction includes:

- **GET MONITOR SWITCHES**
- **UPDATE MONITOR SWITCHES**
- **RESET MONITOR**

This limitation is because such commands use an ATTACH command, while snapshot table functions use CONNECT statement..

## Procedure

- To capture a snapshot using a snapshot administrative view:
  1. Connect to a database. This can be any database in the instance you need to monitor. To be able to issue an SQL query with a snapshot administrative view, you must be connected to a database.
  2. Determine the type of snapshot you need to capture. If you want to capture a snapshot for a database other than the currently connected database, or if you want to retrieve data from a single database partition, or global aggregate data, you need to use a snapshot table function instead.

3. Issue a query with the appropriate snapshot administrative view. For example, here is a query that captures a snapshot of lock information for the currently connected database:

```
SELECT * FROM SYSIBMADM.SNAPLOCK
```

- To capture a snapshot using a snapshot table function:

1. Connect to a database. This can be any database in the instance you need to monitor. To be able to issue an SQL query with a snapshot table function, you must be connected to a database.
2. Determine the type of snapshot you need to capture.
3. Issue a query with the appropriate snapshot table function. For example, here is a query that captures a snapshot of lock information about the SAMPLE database for the current connected database partition:

```
SELECT * FROM TABLE(SNAP_GET_LOCK('SAMPLE',-1)) AS SNAPLOCK
```

The SQL table functions have two input parameters:

**database name**

VARCHAR(255). If you enter NULL, the name of the currently connected database is used.

**partition number**

SMALLINT. For the database partition number parameter, enter the integer (a value between 0 and 999) corresponding to the database partition number you need to monitor. To capture a snapshot for the currently connected database partition, enter a value of -1. To capture a global aggregate snapshot, enter a value of -2. To capture a snapshot from all database partitions, do not specify a value for this parameter.

**Note:**

- a. For the following list of snapshot table functions, if you enter a NULL for the currently connected database, you will get snapshot information for all databases in the instance:
  - SNAP\_GET\_DB
  - SNAP\_GET\_DB\_MEMORY\_POOL
  - SNAP\_GET\_DETAILLOG
  - SNAP\_GET\_HADR
  - SNAP\_GET\_STORAGE\_PATHS
  - SNAP\_GET\_APPL
  - SNAP\_GET\_APPL\_INFO
  - SNAP\_GET\_AGENT
  - SNAP\_GET\_AGENT\_MEMORY\_POOL
  - SNAP\_GET\_STMT
  - SNAP\_GET\_SUBSECTION
  - SNAP\_GET\_BP
  - SNAP\_GET\_BP\_PART
- b. The database name parameter does not apply to the database manager level snapshot table functions; they have only a parameter for database partition number. The database partition number parameter is optional.

## Capturing database system snapshot information to a file using the SNAP\_WRITE\_FILE stored procedure

With the SNAP\_WRITE\_FILE stored procedure you can capture snapshots of monitor data and save this information to files on the database server and allow access to the data by users who do not have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority. Any user can then issue a query with a snapshot table function to access the snapshot information in these files. In providing open access to snapshot monitor data, sensitive information (such as the list of connected users and the SQL statements they have submitted to the database) is available to all users who have the execution privilege for the snapshot table functions. The privilege to execute the snapshot table functions is granted to PUBLIC by default. (Note, however, that no actual data from tables or user passwords can be exposed using the snapshot monitor table functions.)

### Before you begin

You must have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority to capture a database snapshot with the SNAP\_WRITE\_FILE stored procedure.

### About this task

When issuing a call to the SNAP\_WRITE\_FILE stored procedure, in addition to identifying the database and partition to be monitored, you need to specify a *snapshot request type*. Each snapshot request type determines the scope of monitor data that is collected. Choose the snapshot request types based on the snapshot table functions users will need to run. The following table lists the snapshot table functions and their corresponding request types.

Table 113. Snapshot request types

| Snapshot table function    | Snapshot request type |
|----------------------------|-----------------------|
| SNAP_GET_AGENT             | APPL_ALL              |
| SNAP_GET_AGENT_MEMORY_POOL | APPL_ALL              |
| SNAP_GET_APPL              | APPL_ALL              |
| SNAP_GET_APPL_INFO         | APPL_ALL              |
| SNAP_GET_STMT              | APPL_ALL              |
| SNAP_GET_SUBSECTION        | APPL_ALL              |
| SNAP_GET_BP_PART           | BUFFERPOOLS_ALL       |
| SNAP_GET_BP                | BUFFERPOOLS_ALL       |
| SNAP_GET_DB                | DBASE_ALL             |
| SNAP_GET_DETAILLOG         | DBASE_ALL             |
| SNAP_GET_DB_MEMORY_POOL    | DBASE_ALL             |
| SNAP_GET_HADR              | DBASE_ALL             |
| SNAP_GET_STORAGE_PATHS     | DBASE_ALL             |
| SNAP_GET_DBM               | DB2                   |
| SNAP_GET_DBM_MEMORY_POOL   | DB2                   |
| SNAP_GET_FCM               | DB2                   |
| SNAP_GET_FCM_PART          | DB2                   |
| SNAP_GET_SWITCHES          | DB2                   |

Table 113. Snapshot request types (continued)

| Snapshot table function | Snapshot request type |
|-------------------------|-----------------------|
| SNAP_GET_DYN_SQL        | DYNAMIC_SQL           |
| SNAP_GET_LOCK           | DBASE_LOCKS           |
| SNAP_GET_LOCKWAIT       | APPL_ALL              |
| SNAP_GET_TAB            | DBASE_TABLES          |
| SNAP_GET_TAB_REORG      | DBASE_TABLES          |
| SNAP_GET_TBSP           | DBASE_TABLESPACES     |
| SNAP_GET_TBSP_PART      | DBASE_TABLESPACES     |
| SNAP_GET_CONTAINER      | DBASE_TABLESPACES     |
| SNAP_GET_TBSP QUIESCER  | DBASE_TABLESPACES     |
| SNAP_GET_TBSP_RANGE     | DBASE_TABLESPACES     |
| SNAP_GET_UTIL           | DB2                   |
| SNAP_GET_UTIL_PROGRESS  | DB2                   |

## Procedure

1. Connect to a database. This can be any database in the instance you need to monitor. To be able to call a stored procedure, you must be connected to a database.
2. Determine the snapshot request type, and the database and partition you need to monitor.
3. Call the SNAP\_WRITE\_FILE stored procedure with the appropriate parameter settings for the snapshot request type, database, and partition. For example, here is a call that will capture a snapshot of application information about the SAMPLE database for the current connected partition:

```
CALL SNAP_WRITE_FILE('APPL_ALL', 'SAMPLE', -1)
```

The SNAP\_WRITE\_FILE stored procedure has three input parameters:

- a snapshot request type (see Table 113 on page 414, which provides a cross-reference of the snapshot table functions and their corresponding request types)
- a VARCHAR (128) for the database name. If you enter NULL, the name of the currently connected database is used.

**Note:** This parameter does not apply to the database manager level snapshot table functions; they only have parameters for request type and partition number.

- a SMALLINT for the partition number (a value between 0 and 999). For the partition number parameter, enter the integer corresponding to partition number you want to monitor. To capture a snapshot for the currently connected partition, enter a value of -1 or a NULL. To capture a global snapshot, enter a value of -2.

## Results

Once the snapshot data has been saved to a file, all users can issue queries with the corresponding snapshot table functions, specifying (NULL, NULL) as input values for database-level table functions, and (NULL) for database manager level

table functions. The monitor data they receive is pulled from the files generated by the SNAP\_WRITE\_FILE stored procedure.

**Note:** While this provides a means to limit user access to sensitive monitor data, this approach does have some limitations:

- The snapshot monitor data available from the SNAP\_WRITE\_FILE files is only as recent as the last time the SNAP\_WRITE\_FILE stored procedure was called. You can ensure that recent snapshot monitor data is available by making calls to the SNAP\_WRITE\_FILE stored procedure at regular intervals. For instance, on UNIX systems you can set a cron job to do this.
- Users issuing queries with the snapshot table functions cannot identify a database or partition to monitor. The database name and partition number identified by the user issuing the SNAP\_WRITE\_FILE calls determine the contents of the files accessible by the snapshot table functions.
- If a user issues an SQL query containing a snapshot table function for which a corresponding SNAP\_WRITE\_FILE request type has not been run, a direct snapshot is attempted for the currently connected database and partition. This operation is successful only if the user has SYSADM, SYSCTRL, SYSMOINT, or SYSMON authority.

## Accessing database system snapshots using snapshot table functions in SQL queries (with file access)

For every request type that authorized users have called the SNAP\_WRITE\_FILE stored procedure, any user can issue queries with the corresponding snapshot table functions. The monitor data they receive will be retrieved from the files generated by the SNAP\_WRITE\_FILE stored procedure.

### Before you begin

For every snapshot table function with which you intend to access SNAP\_WRITE\_FILE files, an authorized user must have issued a SNAP\_WRITE\_FILE stored procedure call with the corresponding snapshot request types. If you issue an SQL query containing a snapshot table function for which a corresponding SNAP\_WRITE\_FILE request type has not been run, a direct snapshot is attempted for the currently connected database and partition. This operation is successful only if the user has SYSADM, SYSCTRL, SYSMOINT, or SYSMON authority.

### About this task

Users who access snapshot data from SNAP\_WRITE\_FILE files with snapshot table functions cannot identify a database or partition to monitor. The database name and partition number identified by the user issuing the SNAP\_WRITE\_FILE calls determine the contents of the SNAP\_WRITE\_FILE files. The snapshot monitor data available from the SNAP\_WRITE\_FILE files is only as recent as the last time the SNAP\_WRITE\_FILE stored procedure captured snapshots.

### Procedure

1. Connect to a database. This can be any database in the instance you need to monitor. To issue an SQL query with a snapshot table function, you must be connected to a database.
2. Determine the type of snapshot you need to capture.
3. Issue a query with the appropriate snapshot table function. For example, here is a query that will capture a snapshot of table space information:

```
SELECT * FROM TABLE(SNAP_GET_TBSP(CAST(NULL AS VARCHAR(1)),
                                CAST(NULL AS INTEGER))) AS SNAP_GET_TBSP
```

**Note:** You must enter NULL values for the database name and partition number parameters. The database name and partition for the snapshot are determined in the call of the SNAP\_WRITE\_FILE stored procedure. Also, the database name parameter does not apply to the database manager level snapshot table functions; they only have a parameter for partition number. Each snapshot table function returns a table with one or more rows, with each column representing a monitor element. Accordingly, the monitor element column names correlate to the monitor element names.

4. You can also select individual monitor elements from the returned table. For example, the following statement will return only the **agent\_id** monitor element:

```
SELECT agent_id FROM TABLE(
                                SNAP_GET_APPL(CAST(NULL AS VARCHAR(1)),
                                                CAST(NULL AS INTEGER)))
                                as SNAP_GET_APPL
```

## Snapshot monitor SQL Administrative Views

There are a number of different snapshot monitor SQL administrative views available, each returning monitor data about a specific area of the database system. For example, the SYSIBMADM.SNAPBPB SQL administrative view captures a snapshot of buffer pool information. The following table lists each available snapshot monitor administrative view.

Table 114. Snapshot Monitor SQL Administrative Views

| Monitor level    | SQL Administrative Views      | Information returned  |
|------------------|-------------------------------|---|
| Database manager | SYSIBMADM.SNAPDBM             | Database manager level information.   |
| Database manager | SYSIBMADM.SNAPFCM             | Database manager level information regarding the fast communication manager (FCM).  |
| Database manager | SYSIBMADM.SNAPFCM_PART        | Database manager level information for a partition regarding the fast communication manager (FCM).  |
| Database manager | SYSIBMADM.SNAPSWITCHES        | Database manager monitor switch settings.   |
| Database manager | SYSIBMADM.SNAPDBM_MEMORY_POOL | Database manager level information about memory usage.  |
| Database         | SYSIBMADM.SNAPDB              | Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.  |
| Database         | SYSIBMADM.SNAPDB_MEMORY_POOL  | Database level information about memory usage for UNIX platforms only.  |
| Application      | SYSIBMADM.SNAPAPPL            | General application level information for each application that is connected to the database. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set). |
| Application      | SYSIBMADM.SNAPAPPL_INFO       | General application level identification information for each application that is connected to the database.  |

Table 114. Snapshot Monitor SQL Administrative Views (continued)

| Monitor level | SQL Administrative Views        | Information returned  |
|---------------|---------------------------------|---|
| Application   | SYSIBMADM.SNAPLOCKWAIT          | Application level information regarding lock waits for the applications connected to the database.  |
| Application   | SYSIBMADM.SNAPSTMT              | Application level information regarding statements for the applications connected to the database. This includes the most recent SQL statement executed (if the statement switch is set).   |
| Application   | SYSIBMADM.SNAPAGENT             | Application level information regarding the agents associated with applications connected to the database.  |
| Application   | SYSIBMADM.SNAPSUBSECTION        | Application level information regarding the subsections of access plans for the applications connected to the database.   |
| Application   | SYSIBMADM.SNAPAGENT_MEMORY_POOL | Information about memory usage at the agent level.  |
| Table         | SYSIBMADM.SNAPTAB               | Table activity information at the database and application level for each application connected to the database. Table activity information at the table level for each table that <i>was accessed</i> by an application connected to the database. Requires the table switch.          |
| Table         | SYSIBMADM.SNAPTAB_REORG         | Table reorganization information at the table level for each table in the database undergoing reorganization.   |
| Lock          | SYSIBMADM.SNAPLOCK              | Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.   |
| Table space   | SYSIBMADM.SNAPTbsp              | Information about table space activity at the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database. Requires the buffer pool switch. |
| Table space   | SYSIBMADM.SNAPTbsp_PART         | Information about table space configuration.  |
| Table space   | SYSIBMADM.SNAPTbsp_QUIESCER     | Information about quiescers at the table space level.   |
| Table space   | SYSIBMADM.SNAPCONTAINER         | Information about table space container configuration at the table space level.   |
| Table space   | SYSIBMADM.SNAPTbsp_RANGE        | Information about ranges for a table space map.   |
| Buffer pool   | SYSIBMADM.SNAPBP                | Buffer pool activity counters for the specified database. Requires the buffer pool switch.  |
| Buffer pool   | SYSIBMADM.SNAPBP_PART           | Information on buffer size and usage, calculated per partition.   |
| Dynamic SQL   | SYSIBMADM.SNAPDYN_SQL           | Point-in-time statement information from the SQL statement cache for the database.  |
| Database      | SYSIBMADM.SNAPUTIL              | Information about utilities.  |
| Database      | SYSIBMADM.SNAPUTIL_PROGRESS     | Information about the progress of utilities.  |

Table 114. Snapshot Monitor SQL Administrative Views (continued)

| Monitor level | SQL Administrative Views        | Information returned  |
|---------------|---------------------------------|---|
| Database      | SYSIBMADM.SNAPDETAILLOG         | Database level information about log files.   |
| Database      | SYSPROC.ADMIN_GET_STORAGE_PATHS | Returns a list of automatic storage paths for the database that includes file system information for each storage path. |

Before capturing a snapshot, consider if you need information from monitor elements that are under monitor switch control. If a particular monitor switch is off, the monitor elements under its control will not be collected. See the individual monitor elements to determine if an element you need is under switch control.

All snapshot monitoring administrative views and associated table functions use a separate instance connection, which is different from the connection the current session uses. Therefore, an implicit instance attachment might be established, and only default database manager monitor switches are effective. Ineffective monitor switches include any that are turned on or off dynamically from the current session or application.

DB2 Version 9.5 also provides you with a set of administrative views that do not only return values of individual monitor elements, but also return computed values that are commonly required in monitoring tasks. For example, the SYSIBMADM.BP\_HITRATIO administrative view returns calculated values for buffer pool hit ratios, which combine a number of individual monitor elements.

Table 115. Snapshot Monitor SQL Administrative Convenience Views

| SQL Administrative Convenience Views | Information returned  |
|--------------------------------------|---|
| SYSIBMADM.APPLICATIONS               | Information about connected database applications.  |
| SYSIBMADM.APPL_PERFORMANCE           | Information about the rate of rows selected versus the number of rows read by an application.                                     |
| SYSIBMADM.BP_HITRATIO                | Buffer pool hit ratios, including total, data, and index, in the database.  |
| SYSIBMADM.BP_READ_IO                 | Information about buffer pool read performance.   |
| SYSIBMADM.BP_WRITE_IO                | Information about buffer pool write performance.  |
| SYSIBMADM.CONTAINER_UTILIZATION      | Information about table space containers and utilization rates.   |
| SYSIBMADM.LOCKS_HELD                 | Information on current locks held.  |
| SYSIBMADM.LOCKWAITS                  | Information about DB2 agents working on behalf of applications that are waiting to obtain locks.                                  |
| SYSIBMADM.LOG_UTILIZATION            | Information about log utilization for the currently connected database.   |
| SYSIBMADM.LONG_RUNNING_SQL           | Information about the longest running SQL in the currently connected database.  |
| SYSIBMADM.QUERY_PREP_COST            | Information about the time required to prepare different SQL statements.  |
| SYSIBMADM.TBSP_UTILIZATION           | Table space configuration and utilization information.  |
| SYSIBMADM.TOP_DYNAMIC_SQL            | The top dynamic SQL statements sortable by number of executions, average execution time, number of sorts, or sorts per statement. |

## SQL access to database system snapshots

There are two ways to access snapshot monitor data with the snapshot monitor SQL table functions (referred to as *snapshot table functions*): direct access, and file access.

### About this task

#### Direct access

Authorized users can issue queries with snapshot table functions and receive result sets containing monitor data. With this approach, access to snapshot monitor data is only available to users that have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority.

To capture snapshot information using direct access:

1. Optional: Set and check the status of the monitor switches. Refer to “Setting system monitor switches from the CLP” on page 440.
2. Capture database system snapshots using SQL. Refer to “Capturing database system snapshots by using snapshot administrative views and table functions” on page 411.

#### File access

Authorized users call the SNAPSHOT\_FILEW stored procedure, identifying the snapshot request type, and the affected partition and database. The SNAPSHOT\_FILEW stored procedure then saves the monitor data into a file on the database server.

Every request type for which authorized users can call the SNAPSHOT\_FILEW stored procedure,

While this is a safe means of providing all users with access to snapshot monitor data, there are limitations to this approach:

- The snapshot monitor data available from the SNAPSHOT\_FILEW files is only as recent as the last time the SNAPSHOT\_FILEW stored procedure was called. You can ensure that recent snapshot monitor data is available by making calls to the SNAPSHOT\_FILEW stored procedure at regular intervals. For instance, on UNIX operating systems you can set a cron job to do this.
- Users issuing queries with the snapshot table functions cannot identify a database or partition to monitor. The database name and partition number identified by the user issuing the SNAPSHOT\_FILEW calls determine the contents of the files accessible by the snapshot table functions.
- If a user issues an SQL query containing a snapshot table function for which a corresponding SNAPSHOT\_FILEW request type has not been run, a direct snapshot is attempted for the currently connected database and partition. This operation is successful only if the user has SYSADM, SYSCTRL, SYSMANT, or SYSMON authority.

The following tasks are performed by the SYSADM, SYSCTRL, SYSMANT, or SYSMON user who captures database system snapshot information to a file.

### Procedure

1. Find out the needs of users who will issue snapshot requests. Specifically, determine the monitor data they need, the database it is to be collected from, and if the collection needs to be limited to a particular partition.

2. Optional: Set and check the status of the monitor switches. Refer to “Setting system monitor switches from the CLP” on page 440.
3. Capture database system snapshot information to a file. Refer to “Capturing database system snapshot information to a file using the SNAP\_WRITE\_FILE stored procedure” on page 414.

## What to do next

Once the SYSADM, SYSCTRL, SYSMAINT, or SYSMON user has completed the preceding steps, all users can access database system snapshot information using snapshot table functions in SQL queries.

## Capturing a database snapshot from the CLP

You can capture database snapshots from the CLP using the GET SNAPSHOT command. A number of different snapshot request types are available, which can be accessed by specifying certain parameters for the GET SNAPSHOT command.

### Before you begin

You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to capture a database snapshot.

You must have an instance attachment to capture a database snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To obtain a snapshot of a remote instance, you must first attach to that instance.

### Procedure

- Optional: Set and check the status of the monitor switches.
- From the CLP, issue the GET SNAPSHOT command with the required parameters. In the following example, the snapshot captures information for all databases:

```
db2 get snapshot for all databases
```

To capture a database snapshot for a specific database, use the following command:

```
db2 get snapshot for database on db-name
```

where *db-name* is the name of the database you are interested in.

- The following example captures database manager level information:

```
db2 get snapshot for dbm
```

- For partitioned database systems, you can capture a database snapshot specifically for a certain partition, or globally for all partitions. To capture a database snapshot for all applications on a specific partition (for example, partition number 2), issue the following command:

```
db2 get snapshot for all applications at dbpartitionnum 2
```

- To capture a database snapshot for all applications on all partitions, issue the following command:

```
db2 get snapshot for all applications global
```

For global snapshots on partitioned databases, the monitor data from all the partitions is aggregated.

## Snapshot monitor CLP commands

The following table lists all the supported snapshot request types. For certain request types, some information is returned only if the associated monitor switch is set ON. See the individual monitor elements to determine if a required element is under switch control.

Table 116. Snapshot Monitor CLP Commands

| Monitor level    | CLP command   | Information returned   |
|------------------|---|--|
| Connections list | <code>list applications [show detail]</code>                            | Application identification information for all applications currently connected to a database that is managed by the DB2 instance on the partition where snapshot is taken.                      |
| Connections list | <code>list applications for database <i>dbname</i> [show detail]</code> | Application identification information for each application currently connected to the specified database.   |
| Connections list | <code>list dcs applications</code>                                      | Application identification information for all DCS applications currently connected to a database that is managed by the DB2 instance on the partition where snapshot is taken.                  |
| Database manager | <code>get snapshot for dbm</code>                                       | Database manager level information, including instance-level monitor switch settings.  |
| Database manager | <code>get dbm monitor switches</code>                                   | Instance-level monitor switch settings.  |
| Database         | <code>get snapshot for database on <i>dbname</i></code>                 | Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.   |
| Database         | <code>get snapshot for all databases</code>                             | Database level information and counters for each database active on the partition. Information is returned only if there is at least one application connected to the database.                  |
| Database         | <code>list active databases</code>                                      | The number of connections to each active database. Includes databases that were started using the <code>ACTIVATE DATABASE</code> command, but have no connections.                               |
| Database         | <code>get snapshot for dcs database on <i>dbname</i></code>             | Database level information and counters for a specific DCS database. Information is returned only if there is at least one application connected to the database.                                |
| Database         | <code>get snapshot for remote database on <i>dbname</i></code>          | Database level information and counters for a specific federated system database. Information is returned only if there is at least one application connected to the database.                   |
| Database         | <code>get snapshot for all remote databases</code>                      | Database level information and counters for each active federated system database on the partition. Information is returned only if there is at least one application connected to the database. |
| Application      | <code>get snapshot for application applid <i>appl-id</i></code>         | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application      | <code>get snapshot for application agentid <i>appl-handle</i></code>    | Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |

Table 116. Snapshot Monitor CLP Commands (continued)

| Monitor level | CLP command   | Information returned   |
|---------------|---|--|
| Application   | get snapshot for applications on <i>dbname</i>                    | Application level information for each application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | get snapshot for all applications                                 | Application level information for each application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | get snapshot for dcs application applid <i>appl-id</i>            | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | get snapshot for all dcs applications                             | Application level information for each DCS application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | get snapshot for dcs application agentid <i>appl-handle</i>       | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | get snapshot for dcs applications on <i>dbname</i>                | Application level information for each DCS application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | get snapshot for remote applications on <i>dbname</i>             | Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | get snapshot for all remote applications                          | Application level information for each federated system application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Table         | get snapshot for tables on <i>dbname</i>                          | Table activity information at the database and application level for each application connected to the database. Table activity information at the table level for each table that <b>was accessed</b> by an application connected to the database. Requires the table switch. |
| Lock          | get snapshot for locks for application applid <i>appl-id</i>      | List of locks held by the application. Lock wait information requires the lock switch.   |
| Lock          | get snapshot for locks for application agentid <i>appl-handle</i> | List of locks held by the application. Lock wait information requires the lock switch.   |
| Lock          | get snapshot for locks on <i>dbname</i>                           | Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.  |
| Table space   | get snapshot for tablespaces on <i>dbname</i>                     | Information about table space activity for a database. Requires the buffer pool switch. Also included is information about containers, quiescers, and ranges. This information is not under switch control.  |
| Buffer pool   | get snapshot for all bufferpools                                  | Buffer pool activity counters. Requires the buffer pool switch.  |

Table 116. Snapshot Monitor CLP Commands (continued)

| Monitor level | CLP command                                   | Information returned  |
|---------------|---|---|
| Buffer pool   | get snapshot for bufferpools on <i>dbname</i> | Buffer pool activity counters for the specified database. Requires the buffer pool switch.  |
| Dynamic SQL   | get snapshot for dynamic sql on <i>dbname</i> | Point-in-time statement information from the SQL statement cache for the database. The information can also be from a remote data source. |

## Capturing a database snapshot from a client application

You can capture database snapshots using the snapshot monitor API in a C, C++, or a COBOL application. In C and C++ a number of different snapshot request types can be accessed by specifying certain parameters in `db2GetSnapshot()`.

### Before you begin

You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to use the `db2MonitorSwitches` API.

You must have an instance attachment to capture a database snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To obtain a snapshot of a remote instance, you must first attach to that instance.

### Procedure

- Optional: Set and check the status of the monitor switches.
- Include the following DB2 libraries: `sqlmon.h` and `db2ApiDf.h`. These are found in the `include` subdirectory under `sqllib`.
- Set snapshot buffer unit size to 100 KB.
- Declare the `sqlca`, `sqlma`, `db2GetSnapshotData`, and `sqlm_collected` structures. Also, initialize a pointer to contain the snapshot buffer, and establish the buffer's size.

```
#include <db2ApiDf.h>
#include <sqlmon.h>

#define SNAPSHOT_BUFFER_UNIT_SZ 102400

struct sqlma *pRequestedDataGroups;
struct sqlca sqlca;
memset (&sqlca, '\0', sizeof(struct sqlca));
struct sqlm_collected collectedData;
memset (&collectedData, '\0', sizeof(collectedData));
db2GetSnapshotData getSnapshotParam;
memset (&getSnapshotParam, '\0', sizeof(getSnapshotParam));

static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
sqluint32 outputFormat;
char *snapshotBuffer;
```

- Initialize the `sqlma` structure and specify that the snapshot to be captured is of database manager level information.
- Initialize the buffer which is to hold the snapshot output.

```
pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
memset(pRequestedDataGroups, '\0', SQLMASIZE(1));
pRequestedDataGroups->obj_num = 1;
pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;

snapshotBuffer = (char *)malloc(snapshotBufferSize);
memset (snapshotBuffer, '\0', snapshotBufferSize);
```

- Populate the `db2GetSnapshotData` structure with the snapshot request type (from the `sqlma` structure), buffer information, and other information required to capture a snapshot.

```

getSnapshotParam.piSqlmaData = pRequestedDataGroups;
getSnapshotParam.poCollectedData = &collectedData;
getSnapshotParam.poBuffer = snapshotBuffer;
getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_DEFAULT;

```

- Capture the snapshot. Pass the `db2GetSnapshotData` structure, which contains the information necessary to capture a snapshot, as well as a reference to the buffer, where snapshot output is to be directed.

```
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
```

- Include logic to handle buffer overflow. After a snapshot is taken, the `sqlcode` is checked for a buffer overflow. If a buffer overflow occurred the buffer is cleared and reinitialized, and the snapshot is taken again.

```

while (sqlca.sqlcode == 1606)
{
    free(snapshotBuffer);
    snapshotBufferSize = snapshotBufferSize +
        SNAPSHOT_BUFFER_UNIT_SZ;
    snapshotBuffer = (char *)malloc(snapshotBufferSize);
    if (snapshotBuffer == NULL)
    {
        printf("\nMemory allocation error.\n");
        return 1;
    }
    getSnapshotParam.iBufferSize = snapshotBufferSize;
    getSnapshotParam.poBuffer = snapshotBuffer;
    db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}

```

- Process the snapshot monitor data stream.

- Clear the buffer.

```

free(snapshotBuffer);
free(pRequestedDataGroups);

```

## Snapshot monitor API request types

The following table lists all the supported snapshot request types. For certain request types, some information is returned only if the associated monitor switch is set ON. See the individual monitor elements to determine if a required element is under switch control.

Table 117. Snapshot Monitor API Request Types

| Monitor level    | API request type     | Information returned  |
|------------------|----------------------|---|
| Connections list | SQLMA_APPLINFO_ALL   | Application identification information for all applications currently connected to a database that is managed by the DB2 instance on the partition where snapshot is taken. |
| Connections list | SQLMA_DBASE_APPLINFO | Application identification information for each application currently connected to the specified database.  |

Table 117. Snapshot Monitor API Request Types (continued)

| Monitor level    | API request type       | Information returned  |
|------------------|------------------------|---|
| Connections list | SQLMA_DCS_APPLINFO_ALL | Application identification information for all DCS applications currently connected to a database that is managed by the DB2 instance on the partition where snapshot is taken.   |
| Database manager | SQLMA_DB2              | Database manager level information, including instance-level monitor switch settings.   |
| Database         | SQLMA_DBASE            | Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.  |
| Database         | SQLMA_DBASE_ALL        | Database level information and counters for each database active on the partition. The number of connections to each active database. Includes databases that were started using the ACTIVATE DATABASE command, but have no connections. Information is returned only if there is at least one application connected to the database. |
| Database         | SQLMA_DCS_DBASE        | Database level information and counters for a specific DCS database. Information is returned only if there is at least one application connected to the database.   |
| Database         | SQLMA_DCS_DBASE_ALL    | Database level information and counters for each DCS database active on the partition. Information is returned only if there is at least one application connected to the database.   |
| Database         | SQLMA_DBASE_REMOTE     | Database level information and counters for a specific federated system database. Information is returned only if there is at least one application connected to the database.  |
| Database         | SQLMA_DBASE_REMOTE_ALL | Database level information and counters for each active federated system database on the partition. Information is returned only if there is at least one application connected to the database.  |
| Application      | SQLMA_APPL             | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application      | SQLMA_AGENT_ID         | Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application      | SQLMA_DBASE_APPLS      | Application level information for each application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application      | SQLMA_APPL_ALL         | Application level information for each application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |

Table 117. Snapshot Monitor API Request Types (continued)

| Monitor level | API request type          | Information returned  |
|---------------|---------------------------|---|
| Application   | SQLMA_DCS_APPL            | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | SQLMA_DCS_APPL_ALL        | Application level information for each DCS application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | SQLMA_DCS_APPL_HANDLE     | Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | SQLMA_DCS_DBASE_APPLS     | Application level information for each DCS application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Application   | SQLMA_DBASE_APPLS_REMOTE  | Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).   |
| Application   | SQLMA_APPL_REMOTE_ALL     | Application level information for each federated system application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).  |
| Table         | SQLMA_DBASE_TABLES        | Table activity information at the database and application level for each application connected to the database. Table activity information at the table level for each table that <b>was accessed</b> by an application connected to the database. Requires the table switch.          |
| Lock          | SQLMA_APPL_LOCKS          | List of locks held by the application. Lock wait information requires the lock switch.  |
| Lock          | SQLMA_APPL_LOCKS_AGENT_ID | List of locks held by the application. Lock wait information requires the lock switch.  |
| Lock          | SQLMA_DBASE_LOCKS         | Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.   |
| Table space   | SQLMA_DBASE_TABLESPACES   | Information about table space activity at the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database. Requires the buffer pool switch. |
| Buffer pool   | SQLMA_BUFFERPOOLS_ALL     | Buffer pool activity counters. Requires the buffer pool switch.   |
| Buffer pool   | SQLMA_DBASE_BUFFERPOOLS   | Buffer pool activity counters for the specified database. Requires the buffer pool switch.  |
| Dynamic SQL   | SQLMA_DYNAMIC_SQL         | Point-in-time statement information from the SQL statement cache for the database.  |



```

Lock Count                = 1
Hold Count                = 0
Lock Object Name          = 3
Object Type               = Table
Tablespace Name           = USERSPACE1
Table Schema              = DB2ADMIN
Table Name                = STAFF
Mode                      = IX

Lock Name                  = 0x01000000010000000100810056
Lock Attributes           = 0x00000000
Release Flags             = 0x40000000
Lock Count                = 1
Hold Count                = 0
Lock Object Name          = 0
Object Type               = Internal Variation Lock
Mode                      = S

Lock Name                  = 0x4141414141414A48520000000041
Lock Attributes           = 0x00000000
Release Flags             = 0x40000000
Lock Count                = 1
Hold Count                = 0
Lock Object Name          = 0
Object Type               = Internal Plan Lock
Mode                      = S

Lock Name                  = 0x434F4E544F4B4E310000000041
Lock Attributes           = 0x00000000
Release Flags             = 0x40000000
Lock Count                = 1
Hold Count                = 0
Lock Object Name          = 0
Object Type               = Internal Plan Lock
Mode                      = S

```

From this snapshot, you can see that there is currently one application connected to the SAMPLE database, and it is holding five locks.

```

Locks held                = 5
Applications currently connected = 1

```

Note that the time (Status change time) when the Application status became UOW Waiting is returned as Not Collected. This is because the UOW switch is OFF.

The lock snapshot also returns the total time spent so far in waiting for locks, by applications connected to this database.

```

Total wait time (ms)      = 0

```

## Subsection snapshots

On systems that use interpartition parallelism, the SQL compiler partitions the access plan for an SQL statement into subsections. Each subsection is executed by a different DB2 agent (or agents for SMP).

The access plan for an SQL statement generated by the DB2 code generator during compilation can be obtained using the db2expln command. As an example, selecting all the rows from a table that is partitioned across several partitions might result in an access plan having two subsections:

1. Subsection 0, the coordinator subsection, whose role is to collect rows fetched by the other DB2 agents (subagents) and return them to the application.
2. Subsection 1, whose role is to perform a table scan and return the rows to the coordinating agent.

In this simple example, subsection 1 would be distributed across all the database partitions. There would be a subagent executing this subsection on each physical partition of the database partition group to which this table belongs.

The database system monitor allows you to correlate runtime information with the access plan, which is compile time information. With interpartition parallelism, the monitor breaks information down to the subsection level. For example, when the statement monitor switch is ON, a GET SNAPSHOT FOR APPLICATION will return information for each subsection executing on this partition, as well as totals for the statement.

The subsection information returned for an application snapshot includes:

- the number of table rows read/written
- CPU consumption
- elapsed time
- the number of table queue rows sent and received from other agents working on this statement. This allows you to track the execution of a long running query by taking a series of snapshots.
- subsection status. If the subsection is in a WAIT state, because it is waiting for another agent to send or receive data, then the information also identifies the partition or partitions preventing the subsection from progressing in its execution. You may then take a snapshot on these partitions to investigate the situation.

The information logged by a statement event monitor for each subsection after it has finished executing includes: CPU consumption, total execution, time, and several other counters.

## Global snapshots on partitioned database systems

On a partitioned database system, you can use the snapshot monitor to take a snapshot of the current partition, a specified partition, or all partitions. When taking a global snapshot across all the partitions of a partitioned database, data is aggregated before the results are returned.

Data is aggregated for the different element types as follows:

- **Counters, Time, and Gauges**  
Contains the sum of all like values collected from each partition in the instance. For example, GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL would return the number of rows read (rows\_read) from the database for all partitions in the partitioned database instance.
- **Watermarks**  
Returns the highest (for high water) or lowest (for low water) value found for any partition in the partitioned database system. If the value returned is of concern, then snapshots for individual partitions can be taken to determine if a particular partition is over utilized, or if the problem is instance-wide.
- **Timestamp**  
Set to the timestamp value for the partition where the snapshot monitor instance agent is attached. Note that all timestamp values are under control of the timestamp monitor switch.
- **Information**  
Returns the most significant information for a partition that may be impeding work. For example, for the element appl\_status, if the status on one partition

was UOW Executing, and on another partition Lock Wait, Lock Wait would be returned, since it is the state that's holding up execution of the application.

You can also reset counters, set monitor switches, and retrieve monitor switch settings for individual partitions or all partitions in your partitioned database.

**Note:** When taking a global snapshot, if one or more partitions encounter an error, then data is collected from the partitions where the snapshot was successful and a warning (sqlcode 1629) is also returned. If a global get or update of monitor switches, or a counter reset fails on one or more partitions, then those partitions will not have their switches set, or data reset.

## Snapshot monitor self-describing data stream

After you capture a snapshot with the db2GetSnapshot API, the API returns the snapshot output as a self-describing data stream. Figure 6 shows the structure of the data stream and Table 118 on page 432 provides some examples of the logical data groups and monitor elements that might be returned.

**Note:** Descriptive names are used for the identifiers in the examples and tables. These names are prefixed by **SQLM\_ELM\_** in the actual data stream. For example, collected would appear as **SQLM\_ELM\_COLLECTED** in the snapshot monitor output. Types are prefixed with **SQLM\_TYPE\_** in the actual data stream. For example, headers appear as **SQLM\_TYPE\_HEADER** in the data stream.

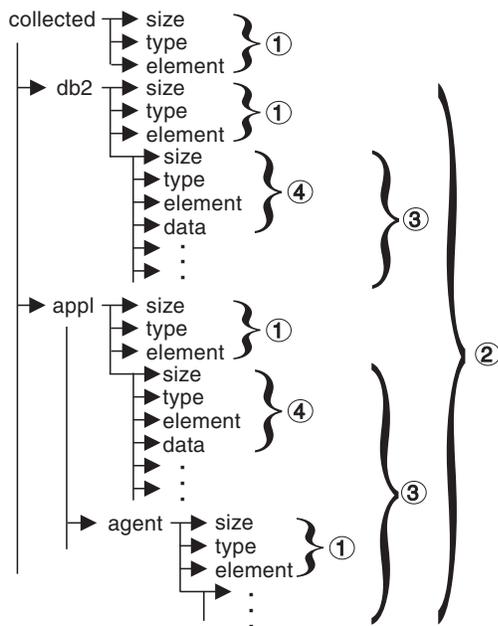


Figure 6. Snapshot Monitor Data Stream

1. Each logical data group begins with a header that indicates its size and name. This size does not include the volume of data taken up by the header itself.
2. Size in the collected header returns the total size of the snapshot.
3. The size element in other headers indicates the size of all the data in that logical data group, including any subordinate groupings.
4. Monitor element information follows its logical data group header and is also self-describing.

Table 118. Sample Snapshot Data Stream

| Logical Data Group | Data Stream                                      | Description  |
|--------------------|--|--|
| collected          | 1000<br>header<br>collected                      | Size of snapshot data (in bytes).<br>Indicates the start of a logical data group.<br>Name of the logical data group.   |
|                    | 4<br>u32bit<br>server_db2_type<br>sqlf_nt_server | Size of the data stored in this monitor element.<br>Monitor element type - unsigned 32 bit numeric.<br>The name of the monitor element collected.<br>The collected value for this element. |
|                    | 2<br>u16bit<br>node_number<br>3                  | Size of the data stored in this monitor element.<br>Monitor element type - unsigned 16 bit numeric.<br>The name of the monitor element collected.<br>The collected value for this element. |
| db2                | 200<br>header<br>db2                             | Size of the DB2 level portion of data in the snapshot.<br>Indicates the start of a logical data group.<br>Name of the logical data group.  |
|                    | 4<br>u32bit<br>sort_heap_allocated<br>16         | Size of the data stored in this monitor element.<br>Monitor element type - unsigned 32 bit numeric.<br>The name of the monitor element collected.<br>The collected value for this element. |
|                    | 4<br>u32bit<br>local_cons<br>3                   | Size of the data stored in this monitor element.<br>Monitor element type - unsigned 32 bit numeric.<br>The name of the monitor element collected.<br>The collected value for this element. |
|                    | ...  | ...  |
| appl               | 100<br>header<br>appl                            | Size of the appl element data in the snapshot.<br>Indicates the start of a logical data group.<br>Name of the logical data group.  |
|                    | 4<br>u32bit<br>locks_held<br>3                   | Size of the data stored in this monitor element.<br>Monitor element type - unsigned 32 bit numeric.<br>The name of the monitor element collected.<br>The collected value for this element. |
|                    | ...  | ...  |

Table 118. Sample Snapshot Data Stream (continued)

| Logical Data Group | Data Stream | Description                                      |
|--------------------|-------------|--|
| agent              | 50          | Size of the agent portion of the appl structure. |
|                    | header      | Indicates the start of a logical data group.     |
|                    | agent       | Name of the logical data group.                  |
| agent              | 4           | Size of the data stored in this monitor element. |
|                    | u32bit      | Monitor element type - 32 bit numeric.           |
|                    | agent_pid   | The name of the monitor element collected.       |
|                    | 12          | The collected value for this element.            |
| ...                | ...         | ...  |

The `db2GetSnapshot()` routine returns the self-describing snapshot data in the user-supplied buffer. Data is returned in the logical data groupings associated with the type of snapshot being captured.

Each item returned by a snapshot request contains fields that specify its size and type. The size can be used to parse through the returned data. A field's size can also be used to skip over a logical data group. For example, to skip over the DB2 record you need to determine the number of bytes in the data stream. Use the following formula to calculate the number of bytes to skip:

$$\text{size of the db2 logical data grouping} + \text{sizeof(sqlm\_header\_info)}$$

## Monitoring with `db2top` in interactive mode commands

The `db2top` monitoring utility quickly and efficiently monitors a complex DB2 environment. It combines DB2 snapshot information from all database partitions and provides a dynamic, real-time view of a running DB2 system using a text-based user interface.

### About this task

When you run `db2top` in interactive mode, you can issue the following commands:

- A** Monitor either the primary or the secondary database in a HADR cluster.
- a** Goto application details for agent (or restrict on agent on statement screen). The `db2top` command will prompt for the agent-id.
- B** Display the main consumer of critical server resources (Bottleneck Analysis).
- c** This option allows to change the order of the columns displayed on the screen. The syntax is in the form: `1,2,3,...` where 1,2,3 correspond to the 1st, 2nd and 3rd columns displayed. These are the column numbers to use when specifying a sort criteria.

When you use the `c` switch key, a screen is shown specifying the order of the columns displayed on screen. The left part of the screen displays the default order and column numbers; the right part of the screen displays the current ordering. To change the order of the columns, enter the new column order in the text field at the bottom of the screen. Next, enter the relative column positions as displayed on the left, separated by commas. Not all columns need to be specified. This column ordering can be saved in `$DB2TOPRC` for subsequent `db2top` monitoring sessions by selecting `w`.

You can sort and select in which order the columns are displayed on the screen. Valid keywords for column ordering in the .db2toprc file are:

- sessions=
- tables=
- tablespaces=
- bufferpools=
- dynsql=
- statements=
- locks=
- utilities=
- federation=

- b** Goto buffer pool screen.
- C** Toggle snapshot data collector on/off.
- d** Goto database screen.
- D** Goto the dynamic SQL screen.
- f** Freeze screen.
- F** Monitor federated queries on the primary server.
- G** Toggle graph on/off.
- h** Go to Help screen
- H** Goto the history screen
- i** Toggle idle sessions on/off.
- k** Toggle actual vs delta values.
- l** Goto sessions screen.
- L** Allows to display the complete query text from the SQL screen. Regular DB2 explain can then be run using e or X options.
- m** Display memory pools.
- o** Display session setup.
- p** Goto the partitions screen.
- P** Select db partition on which to issue snapshot.
- q** Quit db2top.
- R** Reset snapshot data.
- s** Goto the statements screen.
- S** Run native DB2 snapshot.
- t** Goto table spaces screen.
- T** Goto tables screen
- u** Display active utilities and aggregate them across database partitions.
- U** Goto the locks screen.
- V** Set default explains schema.
- w** Write session settings to .db2toprc.
- W** Watch mode for agent\_id, os\_user, db\_user, application or netname.

Statements returned by the session snapshot (option I) will be written to agent.sql, os\_user-agent.sql, db\_user-agent.sql, application-agent.sql or netname-agent.sql. When issued from the dynamic SQL screen (option D), statements will be written to db2adv.sql in a format compatible with db2adviz.

- X Toggle extended mode on/off.
- z|Z Sort on ascending or descending order.
- / Enter expression to filter data. Expression must conform to regular expression. Each function (screen) can be filtered differently. The regexp check is applied to the whole row.
- <l> Move to left or right of screen.

The following switches apply to the applications screen only:

- r Return to previous function.
- R Toggle automatic refresh.
- g Toggle graph on/off.
- X Toggle extended mode on/off.
- d Display agents.

To start **db2top in interactive mode**, issue the following command:

```
db2top -d <database name>
```

When you type

```
db2top -d sample
```

the following output is displayed:

```
[V]11:57:10,refresh=2secs(0.000) Inactive,part=[1/1],<instanceName>:sample
[d=Y,a=N,e=N,p=ALL] [qp=off]

[/]: When rotating, it means that db2top is waiting between two snapshots, otherwise, it means db2top is waiting
      from an answer from DB2
11:57:10: current time
refresh=2secs: time interval
refresh=!secs: Exclamation mark means the time to process the snapshot by DB2 is longer than the refresh interval.
               In this case, db2top will increase the interval by 50%. If this occurs too often because the system is too busy,
               you can either increase the snapshot interval (option I), monitor a single database partition (option P), or turn
               off extended display mode (option x)
0.000 : time spent inside DB2 to process the snapshot
d=Y/N : delta or cumulative snapshot indicator (command option -k or option k).
a=Y/N : Active only or all objects indicator (-a command option set or i)
e=Y/N : Extended display indicator
p=ALL : All database partitions
p=CUR: Current database partition (-P command option with no partition number specified)
p=3 : target database partition number: say 3

Inactive: : Shows inactive if DB2 is not running, otherwise displays the platform on which DB2 is running
part=[1/1] : active database partition number vs total database partition number. For example, part=[2,3] means one
             database partition out of 3 is down (2 active, 3 total)
<instanceName> : instance name
sample : database name
```

## Example

The following example demonstrates running the **db2top** monitoring utility in interactive mode in a partitioned database environment:

```
db2top -d TEST -n mynode -u user -p passwd -V skm4 -B -i 1
```

The command parameters are as follows:

```
-d TEST      # database name
```

```

-n mynode # node name
-u user   # user id
-p passwd # password
-V skm4   # Schema name
-B        # Bold enabled
-i 1      # Screen update interval: 1 second

```

## .db2toprc configuration file

The `.db2toprc` configuration file is a user generated file used to set parameters at initialization time for the **db2top** monitoring utility.

The **db2top** utility will search for the location of the `.db2toprc` file using the user-defined variable `$db2topRC`. If the variable is not set, **db2top** will first search for the `.db2toprc` file in the current directory and then in the home directory. The `.db2toprc` file is user generated.

## Environment variables

You can set the following environment variables:

- **DB2TOPRC**

A user defined environment variable that stores the location of the `.db2toprc` file. For example, on Linux, you can define **DB2TOPRC** as: `export db2topRC=~/.db2toprc`.

If the variable is not set by the user, **db2top** will first search for the `.db2toprc` file in the current directory and then in the home directory.

- **DB2DBDFT**

This variable specifies the database alias name of the database to be used for implicit connects. It is used when no database name is specified on the command line or in the `.db2toprc` configuration file.

- **EDITOR**

This system environment variable specifies the command used to start the text editor used to display the results of explain or native snapshots.

If this variable is not set, **vi** is used.

## Structure

Some of the entries in the `.db2toprc` file are described here.

### cpu=command

Use this entry to display the results of CPU activity on the second line at the right of the screen output. For example:

```
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)", $14+$15);}'
```

displays `Cpu=2(usr+sys)` on the right of the screen.

### io=command

Use this entry to specify a command and display the result on the second line at the left of the screen output. For example:

```
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)", $10+$11);}'
```

displays `Disk=76(bi+bo)` on the left of the screen.

Both commands run as background processes and the fields on the screen are updated asynchronously.

### shell alias=command

Use this shell entry to specify a user defined command, for example: shell M=top spawns top from a **db2top** session when entering M. It returns to the current screen upon exit.

### function alias=command

Use this entry to specify a user defined command, for example: function N=netstat creates a new function called N that repeatedly displays the output of **netstat**. There can be multiple function entries. They must be placed on separate lines. For example:

```
function Q=netstat
function N=df -k
```

### sort=command

Use this entry to specify a sort order, for example: sort=command creates a default sort order for this function, where command is the column number. It can be either ascending or descending. Sort is valid for sessions, tables, tablespace, bufferpool, dynsql, statements, locks, utilities and federation.

## Sample .db2toprc file

There is no default .db2toprc configuration file. However, you can press "W" to create a .db2toprc for the current setup. Use the following sample **.db2toprc** file as a reference. Comments have been added to all entries.

```
# db2top configuration file
# On UNIX, should be located in $HOME/.db2toprc
# File generated by db2top-1.0a
#
node= # [-n] nodename
database=sample # [-d] databasename
user= # [-u] database user
password= # [-p] user password (crypted)
schema= # [-V] default schema for explains
interval=2 # [-i] sampling interval
active=OFF # [-a] display active sessions only (on/off)
reset=OFF # [-R] Reset snapshot at startup (on/off)
delta=ON # [-k] Toggle display of delta/cumulative values (on/off)
gauge=ON # display graph on sessions list (on/off)
colors=ON
# True if terminal supports colors. Informs GE_WRS if it can display information with colors
graphic=ON # True if terminal supports semi graphical characters (on/off).
port= # Port for network collection
streamsize=size # Max collection size per hour (eg. 1024 or 1K : K, M or G)
# Command to get cpu usage information from OS
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)",$14+$15);}'
# Command to get IO usage information from OS
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)",$10+$11);}'
# Ordering of information in sessions screen
# Column order for the session screen (option l)
sessions=0,1,18,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20,21,22,23
# Column order for the tables screen (option T)
tables=0,1,2,4,3,5,6,7
# Column order for the tablespaces screen (option t).
# The display will be sorted in ascending order on column #22
tablespaces=0,1,18,2,3,4,5,6,7,8, sort=22a
# Column order for the bufferpool screen (option b)
bufferpools=0,1,18,2,3,4,5,6,7,8,9,10
# Column order for the Dynamic SQL screen (option D)
dynsql=0,1,18,2,3,4,5,6,7,8,9
statements=0,1
locks=0,1
utilities=0 # contains the default column and sort order for the utility screen
federation=0,2,4 # contains the default column and sort order for the federation screen

# User defined commands
shell P=top
function N=date && netstat -t tcp
```

---

## Switch-based monitoring concepts

### System monitor switches

System monitor switches control how snapshot monitors and some event monitors collect data.

**Note:** These system monitor switches do not affect the unit of work event monitor and locking event monitor, which were introduced in DB2 Version 9.7.

The snapshot monitor and some event monitors report data collected by the system monitor. Collecting system monitor data introduces extra processing for the database manager. For example, in order to calculate the execution time of SQL statements, the database manager must make calls to the operating system to obtain timestamps before and after the execution of every statement. These types of system calls are generally expensive. The system monitor also increases memory consumption. For every monitor element tracked by the system monitor, the database manager uses its memory to store the collected data.

In order to minimize the additional processor time involved in maintaining monitoring information, monitor switches control the collection of potentially expensive data by the database manager. Each switch has only two settings: ON or OFF. If a monitor switch is OFF, the monitor elements under that switch's control do not collect any information. There is a considerable amount of basic monitoring data that is not under switch control, and will always be collected regardless of switch settings.

Each monitoring application has its own logical view of the monitor switches (and the system monitor data). Upon startup each application inherits its monitor switch settings from the `dft_monswitches` parameters in the database manager configuration file (at the instance level). A monitoring application can alter its monitor switch settings with the `UPDATE MONITOR SWITCHES USING MONSWITCH OFF/ON` command. The `MONSWITCH` parameter holds values found in the Monitor Switch column in the Snapshot Monitor Switches table shown in the next section. Changes to the switch settings at the application level only affect the application from where the switch was changed.

Instance-level monitor switches can be changed without stopping the database management system. To do this use the `UPDATE DBM CFG USING DBMSWITCH OFF/ON` command. The `DBMSWITCH` parameter holds values from the DBM Parameter column in the Snapshot Monitor Switches table shown in the next section. This dynamic updating of switches requires that the application performing the update be explicitly attached to the instance for the updates to dynamically take effect. Other existing snapshot applications will not be affected by a dynamic update. New monitoring applications will inherit the updated instance-level monitor switch settings. For an existing monitoring application to inherit the new default monitor switch values, it must terminate and re-establish its attachment. Updating the switches in the database manager configuration file will update the switches for all partitions in a partitioned database.

The database manager keeps track of all the snapshot monitoring applications and their switch settings. If a switch is set to ON in one application's configuration, then the database manager always collects that monitor data. If the same switch is then set to OFF in the application's configuration, then the database manager will still collect data as long as there is at least one application with this switch turned ON.

The collection of time and timestamp elements is controlled by the `TIMESTAMP` switch. Turning this switch OFF (it is ON by default) instructs the database manager to skip any timestamp operating system calls when determining time or timestamp-related monitor elements. Turning this switch OFF becomes important as CPU utilization approaches 100%. When this occurs, the performance degradation caused by issuing timestamps increases dramatically. For monitor elements that can be controlled by the `TIMESTAMP` switch and another switch, if either of the switches is turned OFF, data is not collected. Therefore, if the `TIMESTAMP` switch is turned OFF, the overall cost of data under the control of other monitor switches is greatly reduced.

Event monitors are not affected by monitor switches in the same way as snapshot monitoring applications. When an event monitor is defined, it automatically turns ON the instance level monitor switches required by the specified event types. For example, a deadlock event monitor will automatically turn ON the `LOCK` monitor switch. The required monitor switches are turned ON when the event monitor is activated. When the event monitor is deactivated, the monitor switches are turned OFF.

The `TIMESTAMP` monitor switch is not set automatically by event monitors. It is the only monitor switch that controls the collection of any monitor elements belonging to event monitor logical data groupings. If the `TIMESTAMP` switch is OFF, most of the timestamp and time monitor elements collected by event monitors will not be collected. These elements are still written to the specified table, file, or pipe, but with a value of zero.

*Table 119. Snapshot Monitor Switches*

| Monitor Switch          | DBM Parameter                  | Information Provided                      |
|-------------------------|--------------------------------|---|
| <code>BUFFERPOOL</code> | <code>DFT_MON_BUFPOOL</code>   | Number of reads and writes, time taken    |
| <code>LOCK</code>       | <code>DFT_MON_LOCK</code>      | Lock wait times, deadlocks                |
| <code>SORT</code>       | <code>DFT_MON_SORT</code>      | Number of heaps used, sort performance    |
| <code>STATEMENT</code>  | <code>DFT_MON_STMT</code>      | Start/stop time, statement identification |
| <code>TABLE</code>      | <code>DFT_MON_TABLE</code>     | Measure of activity (rows read/written)   |
| <code>UOW</code>        | <code>DFT_MON_UOW</code>       | Start/end times, completion status        |
| <code>TIMESTAMP</code>  | <code>DFT_MON_TIMESTAMP</code> | Timestamps                                |

Before capturing a snapshot or using an event monitor, you must determine what data you need the database manager to gather. If you want any of the following special types of data to be collected in a snapshot, you will need to set the appropriate monitor switches.

- Buffer pool activity information
- Lock, lock wait, and time related lock information
- Sorting information
- SQL statement information
- Table activity information
- Times and timestamp information

- Unit of work information

The switches corresponding to the previously listed information types are all OFF by default, except for the switch corresponding to times and timestamp information, which is ON by default.

Event monitors are only affected by the time and timestamp information switch. All other switch settings have no effect on the data collected by event monitors.

## Setting system monitor switches from the CLP

System monitor switches control the collection of data by the system monitor. By setting certain monitor switches to ON, you can collect specific types of monitor data.

### Before you begin

The application performing any monitor switch updates must have an instance attachment. You must have one of SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to use the following commands:

- UPDATE MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET DATABASE MANAGER MONITOR SWITCHES

You must have SYSADM authority to use the UPDATE DBM CFG command.

### About this task

#### Procedure

- To activate any of the local monitor switches, use the UPDATE MONITOR SWITCHES command. The switches will remain active until the application (CLP) detaches, or until they are deactivated with another UPDATE MONITOR SWITCHES command. The following example updates all of the local monitor switches to be ON:

```
db2 update monitor switches using BUFFERPOOL on LOCK on
      SORT on STATEMENT on TIMESTAMP on TABLE on UOW on
```

- To deactivate any of the local monitor switches, use the UPDATE MONITOR SWITCHES command. The following example updates all of the local monitor switches to be OFF:

```
db2 update monitor switches using BUFFERPOOL off, LOCK off,
      SORT off, STATEMENT off, TIMESTAMP off, TABLE off, UOW off
```

The following is an example of the output you would expect to see after issuing the previously shown UPDATE MONITOR SWITCH command:

#### Monitor Recording Switches

```
Switch list for db partition number 1
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                (LOCK) = OFF
Sorting Information              (SORT) = OFF
SQL Statement Information        (STATEMENT) = OFF
Table Activity Information       (TABLE) = OFF
Unit of Work Information         (UOW) = OFF
Get timestamp information        (TIMESTAMP) = OFF
```

- It is also possible to manipulate the monitor switches at the database manager level. This involves changing the dft\_monswitches parameters in the database

manager configuration file, using the UPDATE DBM CFG command. In the following example, only lock switch controlled information is to be collected in addition to the basic information.

```
db2 update dbm cfg using DFT_MON_LOCK on
```

Whenever a monitoring application is started, it inherits its monitor switch settings from the database manager. Any changes to the database manager's monitor switch settings will not impact any running monitoring applications. Monitoring applications must reattach themselves to the instance to pick up any changes to monitor switch settings.

- For partitioned database systems, you can set monitor switches specifically for a certain partition, or globally for all partitions.

1. To set a monitor switch (for example, BUFFERPOOL) for a specific partition (for example, partition number 3), issue the following command:

```
db2 update monitor switches using BUFFERPOOL on
    at dbpartitionnum 3
```

2. To set a monitor switch (for example, SORT) for all partitions, issue the following command:

```
db2 update monitor switches using SORT on global
```

- To check the status of the local monitor switches use the GET MONITOR SWITCHES command.

```
db2 get monitor switches
```

- For partitioned database systems, you can view the monitor switch settings specifically for a certain partition, or globally for all partitions.

1. To view the monitor switch settings for a specific partition (for example, partition number 2), issue the following command:

```
db2 get monitor switches at dbpartitionnum 2
```

2. To view the monitor switch settings for all partitions, issue the following command:

```
db2 get monitor switches global
```

- To check the status of the monitor switches at the database manager level (or instance level) use the GET DATABASE MANAGER MONITOR SWITCHES command. This command will show the overall switch settings for the instance being monitored.

```
db2 get database manager monitor switches
```

The following is an example of the output you should expect to see after issuing the previously shown command:

```
DBM System Monitor Information Collected
```

```
Switch list for db partition number 1
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information (LOCK) = ON 10-25-2001 16:04:39
Sorting Information (SORT) = OFF
SQL Statement Information (STATEMENT) = OFF
Table Activity Information (TABLE) = OFF
Unit of Work Information (UOW) = OFF
Get timestamp information (TIMESTAMP) = OFF
```

## Results

Now that you have set the required monitor switches and confirmed the switch settings, you are ready to capture and collect monitor data.

## Setting system monitor switches from a client application

System monitor switches control the collection of data by the system monitor. By setting certain monitor switches to ON, you can collect specific types of monitor data.

### Before you begin

The application performing any monitor switch updates must have an instance attachment. You must have SYSADM, SYSCTRL, SYSMOINT, or SYSMON authority to use the db2MonitorSwitches API.

### Procedure

1. Include the following DB2 libraries: sqlutil.h and db2ApiDf.h. These are found in the include subdirectory under sqllib.

```
#include <sqlutil.h>
#include <db2ApiDf.h>
#include <string.h>
#include <sqlmon.h>
```

2. Set switch lists buffer unit size to 1 KB.

```
#define SWITCHES_BUFFER_UNIT_SZ 1024
```

3. Initialize the sqlca, db2MonitorSwitches, and sqlm\_recording\_group structures. Also, initialize a pointer to contain the switch lists buffer, and establish the buffer's size.

```
struct sqlca sqlca;
memset (&sqlca, '\0', sizeof(struct sqlca));
db2MonitorSwitchesData switchesData;
memset (&switchesData, '\0', sizeof(switchesData));
struct sqlm_recording_group switchesList[SQLM_NUM_GROUPS];
memset(switchesList, '\0', sizeof(switchesList));
sqluint32 outputFormat;
static sqluint32 switchesBufferSize = SWITCHES_BUFFER_UNIT_SZ;
char *switchesBuffer;
```

4. Initialize the buffer, which is to hold the switch list output.

```
switchesBuffer = (char *)malloc(switchesBufferSize);
memset(switchesBuffer, '\0', switchesBufferSize);
```

5. To alter the state of the local monitor switches, alter the elements in the sqlm\_recording\_group structure (named switchesList as indicated in the previous step). For a monitor switch to be turned on, the parameter input\_state is to be set to SQLM\_ON. For a monitor switch to be turned off, the parameter input\_state must be set to SQLM\_OFF.

```
switchesList[SQLM_UOW_SW].input_state = SQLM_ON;
switchesList[SQLM_STATEMENT_SW].input_state = SQLM_ON;
switchesList[SQLM_TABLE_SW].input_state = SQLM_ON;
switchesList[SQLM_BUFFER_POOL_SW].input_state = SQLM_OFF;
switchesList[SQLM_LOCK_SW].input_state = SQLM_OFF;
switchesList[SQLM_SORT_SW].input_state = SQLM_OFF;
switchesList[SQLM_TIMESTAMP_SW].input_state = SQLM_OFF;
switchesData.piGroupStates = switchesList;
switchesData.poBuffer = switchesBuffer;
switchesData.iVersion = SQLM_DBMON_VERSION9_5;
switchesData.iBufferSize = switchesBufferSize;
switchesData.iReturnData = 0;
switchesData.iNodeNumber = SQLM_CURRENT_NODE;
switchesData.poOutputFormat = &outputFormat;
```

**Note:** SQLM\_TIMESTAMP\_SW is unavailable if iVersion is less than SQLM\_DBMON\_VERSION8.

6. To submit the changes to switch settings, call the `db2MonitorSwitches()` function. Pass the `db2MonitorSwitchesData` structure (named `switchesData` in this example) as a parameter to the `db2MonitorSwitches` API. The `switchesData` contains the `sqlm_recording_group` structure as a parameter.
 

```
db2MonitorSwitches(db2Version810, &switchesData, &sqlca);
```
7. Process the switch list data stream from the switch list buffer.
8. Clear the switch list buffer.
 

```
free(switchesBuffer);
free(pRequestedDataGroups);
```

## Results

Now that you have set the required monitor switches and confirmed the switch settings, you are ready to capture and collect monitor data.

### System monitor switches self-describing data stream

After you update or view the current system monitor switch settings with the `db2MonitorSwitches` API, the API returns the switch settings as a self-describing data stream. Figure 7 shows the structure of the switch list information that may be returned for a partitioned database environment.

#### Note:

1. Descriptive names are used for the identifiers in the examples and tables. These names are prefixed by **SQLM\_ELM\_** in the actual data stream. For example, `db_event` would appear as `SQLM_ELM_DB_EVENT` in the event monitor output. Types are prefixed with **SQLM\_TYPE\_** in the actual data stream. For example, headers appear as `SQLM_TYPE_HEADER` in the data stream.
2. For global switch requests the partition order of the returned information can be different in each switch request. In this case, a partition id is included in the data stream.

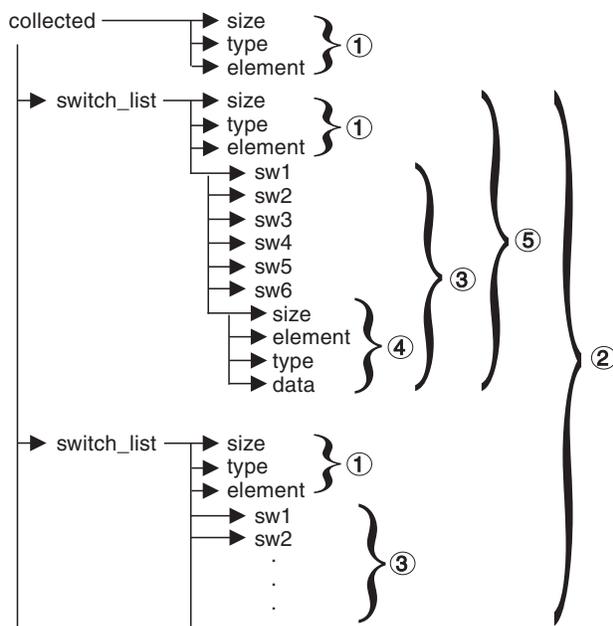


Figure 7. Switch List Monitor Data Stream

1. Each logical data group begins with a header that indicates its size and name. This size does not include the volume of data taken up by the header itself.
2. Size in the collected header returns the total size of all monitor switch lists for all partitions.
3. The size element in switch list header indicates the size of switch data for that partition.
4. Switch information is self-describing.
5. For a non-partitioned database, the switch settings for the stand alone partition are returned. That is, only one switch list is returned.

## Database system monitor data organization

The system monitor collects and stores information that you can access using interfaces to the snapshot monitor and some event monitors. The database system monitor stores information it collects in entities called *monitor elements* (these were previously known as data elements). Each monitor element stores information regarding one specific aspect of the state of the database system.

In addition, monitor elements are identified by unique names and store a certain type of information.

The following element types are available for the system monitor to store data:

### Counter

Counts the number of times an activity occurs. Counter values increase during monitoring. Most counter elements can be reset.

**Gauge** Indicates the current value for an item. Gauge values can go up and down depending on database activity (for example, the number of locks held). Gauge elements can not be reset.

### Watermark

Indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started. Watermark elements can not be reset.

### Information

Provides reference-type details of your monitoring activities. This can include items such as partition names, aliases, and path details. Information elements can not be reset.

### Timestamp

Indicates the date and time that an activity took place by providing the number of seconds and microseconds that have elapsed since January 1, 1970. For the snapshot monitor and event monitors, the collection of timestamp elements is controlled by the `TIMESTAMP` monitor switch. While this switch is on by default, you should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Timestamp elements can not be reset.

A value of 0 for the timestamp element means "not available". If you attempt to import this data, such a value will generate an out of range error (SQL0181). To avoid this error, update the value to any valid timestamp value before exporting the data.

**Time** Returns the number of seconds and microseconds spent on an activity. For the snapshot monitor and event monitors, the collection of most time elements is controlled by the `TIMESTAMP` monitor switch. While this

switch is on by default, you should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Some time elements can be reset.

Monitor elements collect data for one or more logical data groups. A logical data group is a collection of monitor elements that gather database system monitoring information for a specific scope of database activity. Monitor elements are sorted in logical data groups based on the levels of information they provide. For example, while snapshot monitoring, the Total Sort Time monitor element returns database (dbase), application (appl), and statement (stmt) information; hence, it appears in each of the logical data groups listed in parentheses.

Although many monitor elements are used by both the snapshot monitor and event monitors, they each use a distinct set of logical data groups. This is because the scopes of database activity for which you can capture a snapshot differ from those for which you can collect event data. Practically speaking, the overall set of monitor elements accessible from the snapshot monitor is different from those accessible from event monitors.

## Counter status and visibility

Among the monitor elements collected by the system monitor are several accumulating counters. These counters are incremented during the operation of the database or database manager, for example, every time an application commits a transaction.

Counters are initialized when their applicable object becomes active. For instance, the number of buffer pool pages read for a database (a basic monitor element) is set to zero when the database is activated.

Some counters that can be collected by the system monitor are controlled by monitor switches. If a particular monitor switch is off, the monitor elements under its control do not collect data. When a monitor switch is turned on, all the associated counters are reset to zero.

Counters returned by event monitors are reset to zero when the event monitor is activated.

Event monitor counting represents a count since one of the following starting points:

- Event monitor startup, for database, table space, and tables.
- Event monitor startup, for existing connections.
- Application connection, for connections made after the monitor was started.
- Start of the next transaction (unit of work) or statement after the monitor was started.
- Occurrence of a deadlock after the monitor was started.

Each event monitor and any monitoring application (an application using the snapshot monitor APIs) has its own logical view of the system monitor data. This means that when counters are reset or initialized, it only affects the event monitor or application that reset or initialized them. Event monitor counters cannot be reset, except by turning the event monitor off, and then on again. An application taking snapshots can reset its view of the counters at any time by using the RESET MONITOR command.

If you start a statement event monitor after a statement starts, the monitor will start collecting information when the next SQL statement starts. As a result, the event monitor will not return information about statements that the database manager is executing when the monitor was started. This is also true for transaction information.

## System monitor output: the self-describing data stream

Aside from presenting system monitor data on screen or storing it in SQL tables, you can develop a client application to process it. The system monitor returns monitor data via a self-describing data stream for both the snapshot monitor and event monitor.

In a snapshot monitoring application you can call the snapshot APIs to capture a snapshot and then directly process the data stream.

Processing event monitor data is different, in that the event data is sent to the application at the pace database events occur. For a pipe event monitor, the application waits for event data to arrive, and then processes it when it does. For a file event monitor, the application parses event files, thus processing event records in batches.

This self-describing data stream allows you to parse through the returned data one element at a time. This opens up numerous monitoring possibilities, including looking for information regarding a particular application or a specific database state.

The returned monitor data is in the following format:

- size** The size (in bytes) of the data stored in the monitor element or logical data grouping. In the case of a logical data grouping, this is the size of all data in the logical group. For example, the database logical grouping (*db*) contains individual monitor elements (such as *total\_log\_used*) along with other logical data groupings, such as rollforward information (*rollforward*). This does not include the size taken up by the 'size', 'type', and 'element' information.
- type** The type of element stored in the data (for example, variable length string or signed 32 bit numeric value). An element type of *header* refers to a logical data grouping for an element.
- element id**  
The identifier for the monitor element that was captured by the monitor. In the case of a logical data grouping, this is the identifier for the group (for example, *collected*, *dbase*, or *event\_db*).
- data** The value collected by a monitor for a monitor element. In the case of a logical data grouping, the data is composed of the monitor elements belonging to it.

All timestamps in monitor elements are returned in two unsigned 4 byte monitor elements (seconds and microseconds). These represent the number of seconds since January 1, 1970 in GMT time.

The size element of strings in monitor elements represents the actual size of data for the string element. This size does not include a null terminator, as the strings are not null terminated.

## Memory requirements for monitor data

The memory required for monitor data is allocated from the monitor heap. Monitor heap size is controlled by the `mon_heap_sz` database configuration parameter. This parameter has a default value of `AUTOMATIC`, meaning that the monitor heap can increase as needed until the `instance_memory` limit is reached.

If you configure the `mon_heap_sz` parameter manually, consider the following factors:

- The number of monitoring applications
- The number and nature of event monitors
- The monitor switches set
- The level of database activity

Consider increasing the value for the `mon_heap_sz` parameter if monitor commands fail with an `SQLCODE` of `-973`.

The following formula provides an approximation of the number of pages required for the monitor heap:

$$\begin{aligned} & \text{(Memory used by applications} && + \\ & \text{Memory used by event monitors} && + \\ & \text{Memory used by monitoring applications} && + \\ & \text{Memory used by Gateway applications)} && / 4096 \end{aligned}$$

### Memory used by each application

- If the `STATEMENT` switch is off, zero
- If the `STATEMENT` switch is on:
  - Add 400 bytes for each statement being run at the same time. (That is, the number of open cursors that an application might have). This is *not* the cumulative total of statements an application has run.
  - If a partitioned database, add the following for each statement:
    - 200 bytes \* (average # of subsections)
- If the application has issued `sqleseti()` info, add the sizes of the `userid`, `aplname`, `workstation` name and `accounting` string.

### Memory used by each event monitor

For each event monitor of type `ACTIVITIES`:

- 3500 bytes
- If the event monitor is for type `TABLES`, add  $36K * (\text{number of CPU cores} + 1)$
- If the event monitor is for type `FILE` or `PIPE`, add  $2K * (\text{number of CPU cores} + 1)$

If you expect a heavy volume, add 250 megabytes for event records. Otherwise add a fraction that depends on the expected amount of work.

For each event monitor of type `LOCKING` or `UOW`:

- 3500 bytes
- $3K * (\text{number of CPU cores} + 1)$

If you expect a heavy volume, add 250 megabytes for event records. Otherwise add a fraction that depends on the expected amount of work.

For each event monitor of the following type: `DATABASE`, `TABLES`, `TABLESPACES`, `BUFFERPOOLS`, `CONNECTIONS`, `DEADLOCK`:

- 4100 bytes
- 2 \* BUFFERSIZE
- If the event monitor is written to a file, add 550 bytes.
- If the event monitor is for type DATABASE:
  - add 6000 bytes
  - add 100 bytes for each statement in the statement cache
- If the event monitor is for type TABLES:
  - add 1500 bytes
  - add 70 bytes for each table accessed
- If the event monitor is for type TABLESPACES:
  - add 450 bytes
  - add 350 bytes for each table space
- If the event monitor is for type BUFFERPOOLS:
  - add 450 bytes
  - add 340 bytes for each buffer pool
- If the event monitor is for type CONNECTIONS:
  - add 1500 bytes
  - for each connected application:
    - add 750 bytes
  - remember to add the value from “Memory used by each application” on page 447.
- If an event monitor is of type DEADLOCK:
  - and the WITH DETAILS HISTORY is running:
    - add  $X*475$  bytes times the maximum number of concurrent applications you expect to be running, where X is the expected maximum number of statements in your application's unit of work.
  - and the WITH DETAILS HISTORY VALUES is running:
    - also add  $X*Y$  bytes times the maximum number of concurrent applications you expect to be running, where Y is the expected maximum size of parameter values being bound into your SQL statements.

### **Memory used by each monitoring application**

- 250 bytes
- For each database being reset:
  - 350 bytes
  - Add 200 bytes for each REMOTE database.
  - If the SORT switch is on, add 25 bytes.
  - If the LOCK switch is on, add 25 bytes.
  - If the TABLE switch is on:
    - add 600 bytes
    - add 75 bytes per table accessed
  - If the BUFFERPOOL switch is on:
    - add 300 bytes
    - add 250 bytes per table space accessed
    - add 250 bytes per buffer pool accessed
  - If the STATEMENT switch is on:

- add 2100 bytes
- add 100 bytes per statement
- For each application connected to the database:
  - add 600 bytes
  - add 200 bytes for every REMOTE database the application is connected to
  - if the SORT switch is on, add 25 bytes
  - if the LOCK switch is on, add 25 bytes
  - if the BUFFERPOOL switch is on, add 250 bytes
- For each DCS database being reset:
  - add 200 bytes for the database
  - add 200 bytes for each application connected to the database
  - if the STATEMENT switch is ON, Transmission level data must be reset:
    - for each database, add 200 bytes for each transmission level
    - for each application, add 200 bytes for each transmission level

### **Memory used by gateway applications**

- 250 bytes for each host database (even if all switches are off)
- 400 bytes for each application (even if all switches are off)
- If the STATEMENT switch is on:
  - For each application, add 200 bytes for each statement being run at the same time (That is, the number of open cursors that an application might have). This is NOT the cumulative total of statements an application has run.
  - Transmission level data must be accounted for:
    - for each database, add 200 bytes for each transmission level
    - for each application, add 200 bytes for each transmission level
- If the UOW switch is on:
  - add 50 bytes for each application
- For each application using a TMDB (for SYNCPOINT TWOPHASE activity):
  - add 20 bytes plus the size of the XID itself
- For any application that has issued sqleseti to set client name, app name, wkstn or accounting:
  - add 800 bytes plus the size of the accounting string itself

## **Monitoring buffer pool activity**

The database server reads and updates all data from a buffer pool. Data is copied from disk to a buffer pool as it is required by applications.

Pages are placed in a buffer pool:

- by the agent. This is synchronous I/O.
- by the I/O servers (prefetchers). This is asynchronous I/O.

Pages are written to disk from a buffer pool:

- by the agent, synchronously
- by page cleaners, asynchronously

If the server needs to read a page of data, and that page is already in the buffer pool, then the ability to access that page is much faster than if the page had to be

read from disk. It is desirable to hit as many pages as possible in the buffer pool. Avoiding disk I/O is an important factor in database performance, therefore proper configuration of the buffer pools is one of the most important considerations for performance tuning.

The buffer pool *hit ratio* indicates the percentage of time that the database manager did not need to load a page from disk in order to service a page request because the page was already in the buffer pool. The greater the buffer pool hit ratio, the lower the frequency of disk I/O.

**Note:** The information that follows discusses buffer pools in environments other than DB2 pureScale environments. Buffer pools work differently in DB2 pureScale environments. For more information, see “Buffer pool monitoring in a DB2 pureScale environment”, in the *Database Monitoring Guide and Reference*.

For example, the overall buffer pool hit ratio can be calculated as follows:

$$\frac{((\text{pool\_data\_lbp\_pages\_found} + \text{pool\_index\_lbp\_pages\_found} + \text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_data\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}))}{(\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads} + \text{pool\_xda\_l\_reads} + \text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads} + \text{pool\_temp\_index\_l\_reads})} \times 100$$

This calculation takes into account all of the pages (index and data) that are cached by the buffer pool.

You can also use the BP\_HITRATIO administrative view as a convenient method of monitoring the hit ratio for your buffer pools.

For a large database, increasing the buffer pool size may have minimal effect on the buffer pool hit ratio. Its number of data pages may be so large, that the statistical chances of a hit are not improved by increasing its size. Instead, you might find that tuning the index buffer pool hit ratio achieves the required result. This can be achieved using two methods:

1. Split the data and indexes into two different buffer pools and tune them separately.
2. Use one buffer pool, but increase its size until the index hit ratio stops increasing. The index buffer pool hit ratio can be calculated as follows:

$$\frac{((\text{pool\_index\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_temp\_index\_l\_reads}))}{\text{pool\_index\_l\_reads}} \times 100$$

The first method is often more effective, but because it requires indexes and data to reside in different table spaces, it may not be an option for existing databases. It also requires tuning two buffer pools instead of one, which can be a more difficult task, particularly when memory is constrained.

You should also consider the impact that prefetchers may be having on the hit ratio. Prefetchers read data pages into the buffer pool anticipating their need by an application (asynchronously). In most situations, these pages are read just before they are needed (the required case). However, prefetchers can cause unnecessary I/O by reading pages into the buffer pool that will not be used. For example, an application starts reading through a table. This is detected and prefetching starts, but the application fills an application buffer and stops reading. Meanwhile,

prefetching has been done for a number of additional pages. I/O has occurred for pages that will not be used and the buffer pool is partially taken up with those pages.

Page cleaners monitor the buffer pool and asynchronously write pages to disk. Their goals are:

- Ensure that agents will always find free pages in the buffer pool. If an agent does not find free pages in the buffer pool, it must clean them itself, and the associated application will have a poorer response.
- Speed database recovery, if a system crash occurs. The more pages that have been written to disk, the smaller the number of log file records that must be processed to recover the database.

Although dirty pages are written out to disk, the pages are not removed from the buffer pool right away, unless the space is needed to read in new pages.

**Note:** Buffer pool information is typically gathered at a table space level, but the facilities of the database system monitor can roll this information up to the buffer pool and database levels. Depending on your type of analysis, you may need to examine this data at any or all of these levels.

## Database system monitor interfaces

| Monitoring task                                 | API                |
|---|--------------------|
| Capturing a snapshot                            | db2GetSnapshot     |
| Converting the self-describing data stream      | db2ConvMonStream   |
| Displaying the database system monitor switches | db2MonitorSwitches |
| Estimating the size of a snapshot               | db2GetSnapshotSize |
| Get/update monitor switches                     | db2MonitorSwitches |
| Resetting monitor counters                      | db2ResetMonitor    |
| Updating the database system monitor switches   | db2MonitorSwitches |

| Monitoring task  | CLP Command                           |
|--|---------------------------------------|
| Capturing a snapshot   | GET SNAPSHOT                          |
| Displaying the database manager monitor switches                         | GET DATABASE MANAGER MONITOR SWITCHES |
| Displaying the monitoring application's monitor switches                 | GET MONITOR SWITCHES                  |
| Formatting the event monitor trace                                       | db2evmon                              |
| Generating sample SQL for write-to-table CREATE EVENT MONITOR statements | db2evtbl                              |
| Listing the active databases   | LIST ACTIVE DATABASES                 |
| Listing the applications connected to a database                         | LIST APPLICATIONS                     |
| Listing the DCS applications   | LIST DCS APPLICATIONS                 |
| Resetting monitor counters   | RESET MONITOR                         |
| Updating the database system monitor switches                            | UPDATE MONITOR SWITCHES               |

| Monitoring task               | SQL Statement           |
|-------------------------------|-------------------------|
| Activating an event monitor   | SET EVENT MONITOR STATE |
| Creating an event monitor     | CREATE EVENT MONITOR    |
| Deactivating an event monitor | SET EVENT MONITOR STATE |
| Removing an event monitor     | DROP                    |
| Writing event monitor values  | FLUSH EVENT MONITOR     |

| Monitoring task   | SQL Function  |
|---|---|
| Determining the state of an event monitor                                 | EVENT_MON_STATE scalar function   |
| Getting a database manager level snapshot                                 | SNAPDBM administrative view and SNAP_GET_DBM table function                     |
| Getting the current monitor switch settings at the database manager level | SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function           |
| Getting a fast communication manager snapshot                             | SNAPFCM administrative view and SNAP_GET_FCM table function                     |
| Getting a fast communication manager snapshot for a given partition       | SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function           |
| Getting a database level snapshot   | SNAPDB administrative view and SNAP_GET_DB table function                       |
| Getting an application level snapshot                                     | SNAPAPPL administrative view and SNAP_GET_APPL table function                   |
| Getting an application level snapshot                                     | SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function         |
| Getting an application level snapshot for lock wait information           | SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function           |
| Getting an application level snapshot for statement information           | SNAPSTMT administrative view and SNAP_GET_STMT table function                   |
| Getting an application level snapshot for agent information               | SNAPAGENT administrative view and SNAP_GET_AGENT table function                 |
| Getting an application level snapshot for subsection information          | SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function       |
| Getting a buffer pool level snapshot                                      | SNAPBP administrative view and SNAP_GET_BP table function                       |
| Getting a table space level snapshot                                      | SNAPTbsp administrative view and SNAP_GET_TBSP table function                   |
| Getting a table space level snapshot for configuration information        | SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART table function         |
| Getting a table space level snapshot for container information            | SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function         |
| Getting a table space level snapshot for quiescer information             | SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function |
| Getting a table space level snapshot for the ranges of a table space map  | SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function       |
| Getting a table level snapshot  | SNAPTAB administrative view and SNAP_GET_TAB table function                     |
| Getting a lock level snapshot   | SNAPLOCK administrative view and SNAP_GET_LOCK table function                   |
| Getting a snapshot of SQL statement cache information                     | SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL table function             |

---

## Determining the date a database object was last used

The last date that an object was used is indicated by the last referenced date (also referred to as the last used date). The last referenced date is available for indexes, packages, tables, table data partitions, and materialized query tables (MQTs). You can use the last referenced date to identify objects which have not been used for an extended period of time and which might be considered as candidates for removal.

The last referenced date is stored in the LASTUSED column of the corresponding catalog table for the object and accessible through the catalog view on the table. Usage information in the catalogs is updated by an engine dispatchable unit (EDU), called **db2lused** (the LASTUSED daemon), that runs on the database catalog partition. Every 15 minutes, the LASTUSED daemon gathers usage information for all objects across all partitions and updates the LASTUSED column in the corresponding catalog tables to write the information to disk. At most, the catalog entry for a given object is updated once per day, which means the same object will not be checked again until a 24 hour interval has passed. The 15 minute interval was chosen to minimally affect performance on the database server and is not user configurable. The updates to the last referenced date are performed asynchronously and, therefore, object access is not immediately recorded in the catalogs.

**Note:** If the corresponding row in a catalog table is locked, an update of usage information might be deferred until the next 15 minute collection interval. Also, when a database is deactivated, any usage information that was not gathered by the LASTUSED daemon before deactivation (for example, any objects accessed for the first time since the last poll was done by the daemon) cannot be written to disk. Explicitly activate the database for this feature to behave as expected.

The last referenced date is of interest when an object has not been used for an extended period of time (for example, several months). The last referenced date is useful in the following cases:

- Tables and table data partitions: can help to identify opportunities to reclaim unused space
- Indexes: can help to identify opportunities to reclaim unused space, avoid unnecessary inserts and maintenance, and can improve compile time by reducing the number of choices for an index to consider
- Packages: can help to detect unused package versions which can be freed
- MQTs: can help to detect unused MQTs, to reclaim unused space, or help to investigate and understand why an MQT is not being used

The following examples describe some specific scenarios in which the last referenced date can be useful:

- To identify opportunities to save space and maintenance time, you can examine last used information for indexes every year by checking the LASTUSED column in the SYSCAT.INDEXES catalog view. If an index has not been used in the last year, the index can be considered as a candidate for being dropped. The final decision to drop an index remains under your control because there might be circumstances in which dropping an index is not required. For example, you might have a table which is known to be accessed only under emergency or infrequent cases where fast access is critical, or the index for a table might be unique and used to enforce the uniqueness constraint even though it is never explicitly used. The last used date information can be used as an aid in making decisions to remove indexes.

- Your company has internal applications that were deployed on the database and were either replaced or are no longer in use after a period of months or years. The retired applications have been identified as opportunities to save space. The last used date information can be used to identify database objects that are no longer in use and were not cleaned up after an application was retired. For example, these database objects might be tables storing values used to populate a GUI. The last used date for these tables can be found in the LASTUSED column of the SYSCAT.TABLES catalog view and this date can be used as a starting point in the investigation of table objects that can be removed to reclaim space.

For additional information about the LASTUSED column of the catalog view for a specific database object, particularly which operations result in an update, see the following topics:

- SYSCAT.DATAPARTITIONS catalog view
- SYSCAT.INDEXES catalog view
- SYSCAT.PACKAGES catalog view
- SYSCAT.TABLES catalog view

---

## Chapter 5. Deprecated monitoring tools

The health monitor and Windows Management Instrumentation (WMI) are deprecated and might be removed in a future release.

Start to use IBM InfoSphere Optim tools. IBM InfoSphere Optim Performance Manager provides a Web interface that you can use to isolate and analyze typical database performance problems. You can also view a summary of the health of your databases and drill down. For more details, see *Monitoring with Optim Performance Manager* at [http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p\\_monitor.html](http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p_monitor.html).

---

### Introduction to the health monitor

The health monitor is a server-side tool that adds a management-by-exception capability by constantly monitoring the health of an instance and active databases. The health monitor can also alert a database administrator (DBA) of potential system health issues.

The health monitor proactively detects issues that might lead to hardware failure, or to unacceptable system performance or capability. The proactive nature of the health monitor enables users to address an issue before it becomes a problem that affects system performance.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

The health monitor checks the state of your system using health indicators to determine whether an alert should be issued. Preconfigured actions can be taken in response to alerts. The health monitor can also log alerts in the administration notification log and send notifications by email or pager. This management-by-exception model frees up valuable DBA resources by generating alerts to potential system health issues without requiring active monitoring.

The health monitor periodically gathers information about the health of the system with a minimal impact to overall performance. It does not turn on any snapshot monitor switches to collect information.

### Health indicators

The health monitor uses health indicators to evaluate the health of specific aspects of database manager performance or database performance. A health indicator measures the health of some aspect of a particular class of database objects, such as table spaces. Criteria are applied to the measurement to determine healthiness. The criteria applied depends on the type of health indicator. A determination of unhealthiness is based on the criteria generates an alert.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information,

see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

Three types of health indicators are returned by the health monitor:

- **Threshold-based** indicators are measurements that represent a statistic (on a continuous range of values) of the behavior of the object. Warning and alarm threshold values define boundaries or zones for normal, warning, and alarm ranges. Threshold-based health indicators have three valid states: Normal, Warning, or Alarm.
- **State-based** indicators are measurements that represent a finite set of two or more distinct states of an object that defines whether the database object or resource is operating normally. One of the states is normal and all others are considered non-normal. State-based health indicators have two valid states: Normal, Attention.
- **Collection state-based** indicators are database-level measurements that represent an aggregate state or one or more objects within the database. Data is captured for each object in the collection and the highest severity of conditions among those objects is represented in the aggregated state. If one or more objects in the collection are in a state requiring an alert, the health indicator shows Attention state. Collection state-based health indicators have two valid states: Normal, Attention.

Health indicators exist at the instance, database, table space, and table space container level.

You can access health monitor information through the CLP, or APIs. You can configure health indicators through these same tools.

An alert is generated in response to either a change from a normal to a non-normal state or a change in the health indicator value to a warning or alarm zone that is based on defined threshold boundaries. There are three types of alerts: attention, warning, and alarm.

- For health indicators measuring distinct states, an attention alert is issued if a non-normal state is registered.
- For health indicators measuring a continuous range of values, threshold values define boundaries or zones for normal, warning and alarm states. For example, if the value enters the threshold range of values that defines an alarm zone, an alarm alert is issued to indicate that the problem needs immediate attention.

The health monitor will only send notification and run an action on the first occurrence of a particular alert condition for a given health indicator. If the health indicator stays in a particular alert condition, no further notification will be sent and no further actions will be run. If the health indicator changes alert conditions, or goes back to normal state and re-enters the alert condition, notification will be sent anew and actions will be run.

The following table shows an example of a health indicator at different refresh intervals and the health monitor response to the health indicator state. This example uses the default warning of 80% and alarm thresholds of 90%.

Table 120. Health indicator conditions at different refresh intervals

| Refresh interval | Value of ts.ts_util (Table space utilization) health indicator | State of ts.ts_util health indicator | Health monitor response  |
|------------------|--|--------------------------------------|--|
| 1                | 80   | warning                              | notification of warning is sent, actions for a warning alert condition are run |
| 2                | 81   | warning                              | no notification is sent, no actions are run                                    |
| 3                | 75   | normal                               | no notification is sent, no actions are run                                    |
| 4                | 85   | warning                              | notification of warning is sent, actions for a warning alert condition are run |
| 5                | 90   | alarm                                | notification of alarm is sent, actions for an alarm condition are run          |

### Health indicator process cycle

The following diagram illustrates the evaluation process for health indicators. The set of steps runs every time the refresh interval for the specific health indicator elapses.

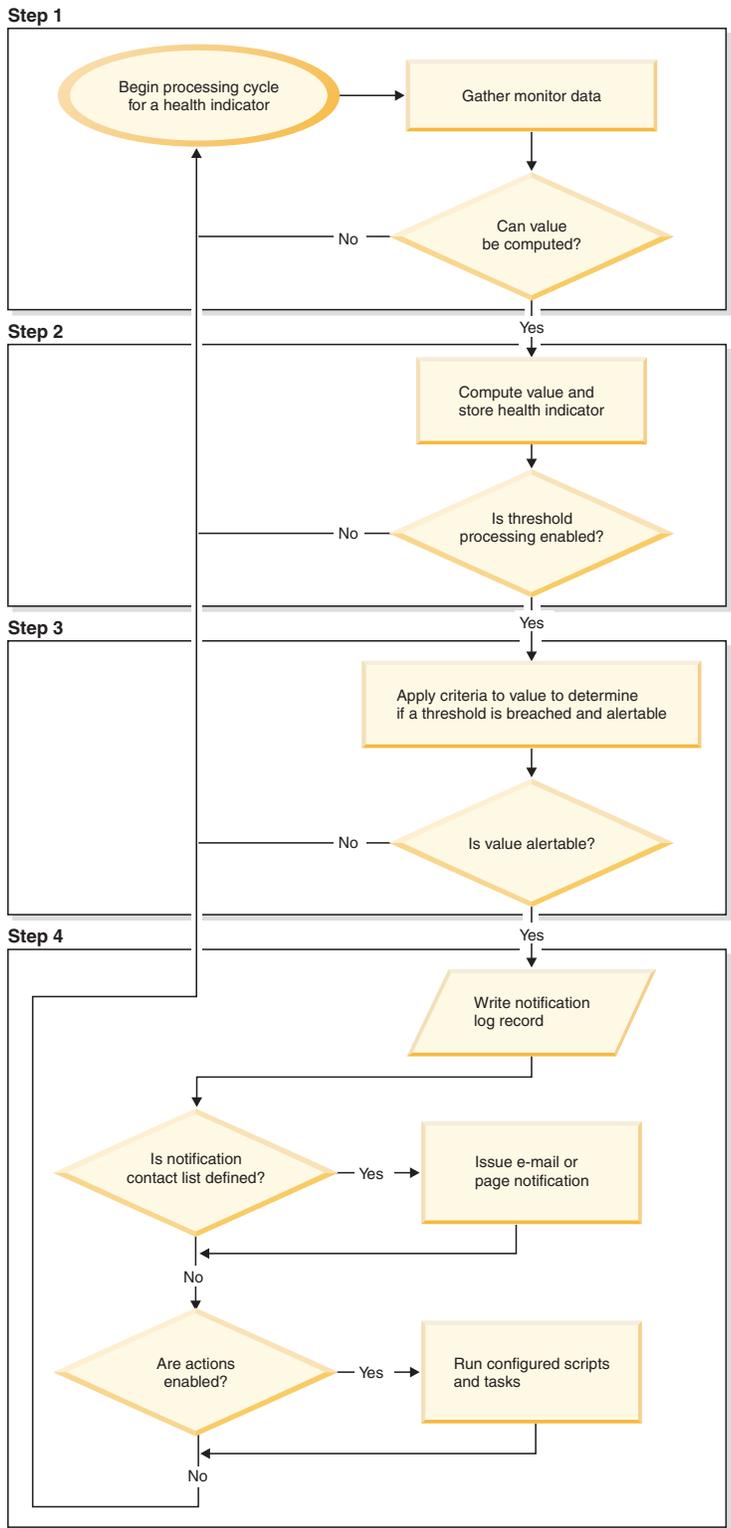


Figure 8. Health indicator process cycle

**Note:**

1. The NOTIFYLEVEL database manager configuration parameter controls whether alert notifications are sent to the DB2 administration notification log

and to any defined contacts. A minimum severity level of 2 is required for alarm notifications. A minimum severity level of 3 is required for warnings and attention alerts to be sent.

### Health indicator format

A description of the data collected by the health indicator.

The documentation for health indicators is described in a standard format as follows:

#### Identifier

The name of the health indicator. This identifier is used for configuration from the CLP.

#### Health monitor level

The level at which the health indicator is captured by the health monitor.

#### Category

The category for the health indicator.

**Type** The type of the health indicator. There are four possible values for type:

- Upper-bounded threshold-based, where the progression to an alert is: Normal, Warning, Alarm
- Lower-bounded threshold-based
- State-based, where one state is normal and all others are non-normal
- Collection state-based, where the state is based on the aggregation of states from objects in the collection

**Unit** The unit of the data measured in the health indicator, such as percentage. This is not applicable for state-based or collection state-based health indicators.

### Health indicators summary

The following tables list all health indicators, grouped by category.

*Table 121. Database automatic storage utilization health indicators*

| Name                                   | Identifier           | Additional Information   |
|--|----------------------|--|
| Database Automatic Storage Utilization | db.auto_storage_util | "db.auto_storage_util - Database automatic storage utilization health indicator" on page 463 |

*Table 122. Table space storage health indicators*

| Name                                     | Identifier               | Additional Information   |
|--|--------------------------|--|
| Table Space Automatic Resize Status      | ts.ts_auto_resize_status | "ts.ts_auto_resize_status - Table space automatic resize status health indicator" on page 463    |
| Automatic Resize Table Space Utilization | ts.ts_util_auto_resize   | "ts.ts_util_auto_resize - Automatic resize table space utilization health indicator" on page 464 |
| Table Space Utilization                  | ts.ts_util               | "ts.ts_util - Table Space Utilization" on page 465   |
| Table Space Container Utilization        | tsc.tscont_util          | "tsc.tscont_util - Table Space Container Utilization" on page 466                                |
| Table Space Operational State            | ts.ts_op_status          | "ts.ts_op_status - Table Space Operational State" on page 467                                    |

Table 122. Table space storage health indicators (continued)

| Name                                    | Identifier               | Additional Information  |
|---|--------------------------|---|
| Table Space Container Operational State | tsc.tscont_op_status     | "tsc.tscont_op_status - Table Space Container Operational State" on page 467                  |
| Table Space Automatic Resize Status     | ts.ts_auto_resize_status | "ts.ts_auto_resize_status - Table space automatic resize status health indicator" on page 463 |

Table 123. Sorting health indicators

| Name                                     | Identifier              | Additional Information   |
|--|-------------------------|--|
| Private Sort Memory Utilization          | db2.sort_privmem_util   | "db2.sort_privmem_util - Private Sort Memory Utilization" on page 467            |
| Shared Sort Memory Utilization           | db.sort_shrmem_util     | "db.sort_shrmem_util - Shared Sort Memory Utilization" on page 468               |
| Percentage of Sorts That Overflowed      | db.spilled_sorts        | "db.spilled_sorts - Percentage of Sorts That Overflowed" on page 469             |
| Long Term Shared Sort Memory Utilization | db.max_sort_shrmem_util | "db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization" on page 470 |

Table 124. Database manager health indicators

| Name                                  | Identifier        | Additional Information                                       |
|---------------------------------------|-------------------|--|
| Instance Operational State            | db2.db2_op_status | "db2.db2_op_status - Instance Operational State" on page 470 |
| Instance Highest Severity Alert State | -                 | "Instance Highest Severity Alert State" on page 471          |

Table 125. Database health indicators

| Name                                  | Identifier      | Additional Information                                     |
|---------------------------------------|-----------------|--|
| Database Operational State            | db.db_op_status | "db.db_op_status - Database Operational State" on page 472 |
| Database Highest Severity Alert State | -               | "Database Highest Severity Alert State" on page 472        |

Table 126. Maintenance health indicators

| Name  | Identifier         | Additional Information  |
|---|--------------------|---|
| Reorganization Required                         | db.tb_reorg_req    | "db.tb_reorg_req - Reorganization Required" on page 473           |
| Statistics Collection Required health indicator | db.tb_runstats_req | "db.tb_runstats_req - Statistics Collection Required" on page 473 |
| Database Backup Required                        | db.db_backup_req   | "db.db_backup_req - Database Backup Required" on page 474         |

Table 127. High availability disaster recovery health indicators

| Name                                     | Identifier        | Additional Information                                    |
|--|-------------------|---|
| HADR Operational Status health indicator | db.hadr_op_status | "db.hadr_op_status - HADR Operational Status" on page 475 |

Table 127. High availability disaster recovery health indicators (continued)

| Name                            | Identifier    | Additional Information                       |
|---------------------------------|---------------|--|
| HADR Log Delay health indicator | db.hadr_delay | "db.hadr_delay - HADR Log Delay" on page 475 |

Table 128. Logging health indicators

| Name                        | Identifier     | Additional Information                                     |
|-----------------------------|----------------|--|
| Log Utilization             | db.log_util    | "db.log_util - Log Utilization" on page 476                |
| Log File System Utilization | db.log_fs_util | "db.log_fs_util - Log File System Utilization" on page 476 |

Table 129. Application concurrency health indicators

| Name  | Identifier            | Additional Information  |
|---|-----------------------|---|
| Deadlock Rate                               | db.deadlock_rate      | "db.deadlock_rate - Deadlock Rate" on page 477                                    |
| Lock List Utilization                       | db.locklist_util      | "db.locklist_util - Lock List Utilization" on page 478                            |
| Lock Escalation Rate                        | db.lock_escal_rate    | "db.lock_escal_rate - Lock Escalation Rate" on page 478                           |
| Percentage of Applications Waiting on Locks | db.apps_waiting_locks | "db.apps_waiting_locks - Percentage of Applications Waiting on Locks" on page 479 |

Table 130. Package cache, catalog cache, and workspace health indicators

| Name                       | Identifier               | Additional Information  |
|----------------------------|--------------------------|---|
| Catalog Cache Hit Ratio    | db.catcache_hitratio     | "db.catcache_hitratio - Catalog Cache Hit Ratio" on page 480        |
| Package Cache Hit Ratio    | db.pkgcache_hitratio     | "db.pkgcache_hitratio - Package Cache Hit Ratio" on page 480        |
| Shared Workspace Hit Ratio | db.shrworkspace_hitratio | "db.shrworkspace_hitratio - Shared Workspace Hit Ratio" on page 481 |

Table 131. Memory health indicators

| Name                      | Identifier        | Additional Information                                     |
|---------------------------|-------------------|--|
| Monitor Heap Utilization  | db2.mon_heap_util | "db2.mon_heap_util - Monitor Heap Utilization" on page 482 |
| Database Heap Utilization | db.db_heap_util   | "db.db_heap_util - Database Heap Utilization" on page 482  |

Table 132. Federated health indicators

| Name                      | Identifier                 | Additional Information   |
|---------------------------|----------------------------|--|
| Nickname Status           | db.fed_nicknames_op_status | "db.fed_nicknames_op_status - Nickname Status" on page 483         |
| Data Source Server Status | db.fed_servers_op_status   | "db.fed_servers_op_status - Data Source Server Status" on page 483 |

## Table space storage health indicators:

*Health indicators for DMS table spaces:*

You can determine the health indicators that are relevant for a DMS table space based on the characteristics of the table space.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

This table describes which table space health indicators are relevant for a DMS table space based on the characteristics of the table space:

*Table 133. Relevant table space health indicators for a DMS table space*

| Table space characteristics    | Maximum table space size defined   | Maximum table space size undefined   |
|--------------------------------|--|--|
| Automatic resize enabled = Yes | <p>ts.ts_util_auto_resize - Tracks percentage of table space used relative to the maximum defined by you. An alert indicates that the table space will soon be full and requires intervention by you. As long as the maximum size has been set to a reasonable value (that is, the amount of space specified by the maximum size does exist), this is the most important health indicator for this configuration.</p> <p>ts.ts_util - Tracks usage of currently allocated table space storage. An alert may not require intervention by you to resolve any problems since the table space will attempt to increase in size when it is full.</p> <p>ts.ts_auto_resize_status - Tracks health of resize attempts. An alert indicates that the table space failed to resize (that is, the table space is full).</p> | <p>ts.ts_util_auto_resize - Not applicable. No upper bound specified for the table space size.</p> <p>ts.ts_util - Tracks usage of currently allocated table space storage. An alert may not require intervention by you to resolve any problems since the table space will attempt to increase in size.</p> <p>ts.ts_auto_resize_status - Tracks health of resize attempts. An alert indicates that the table space failed to resize (that is, the table space is full). <b>Note:</b> If a DMS table space is defined using automatic storage and there is no maximum size specified, you should also pay attention to the db.auto_storage_util health indicator. This health indicator tracks utilization of the space associated with the database storage paths. When this space fills up, the table space is unable to grow. This may result in a table space full condition.</p> |
| Automatic resize enabled = No  | Not a valid configuration. Maximum table space size is only valid for table spaces that have automatic resize enabled.   | <p>ts.ts_util_auto_resize - Not applicable. Table space will not attempt to resize.</p> <p>ts.ts_util - Tracks usage of currently allocated table space storage. An alert indicates a table space full condition and requires immediate intervention by you. The table space will not attempt to resize itself.</p> <p>ts.ts_auto_resize_status - Not applicable. Table space will not attempt to resize.</p>  |

*db.auto\_storage\_util* - Database automatic storage utilization health indicator:

Indicates the consumption of storage for the defined database storage paths.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.auto\_storage\_util

**Health monitor level**

Database

**Category**

Database

**Type** Upper-bounded threshold-based

**Unit** Percentage

When automatic storage table spaces are created, containers are allocated automatically for these table spaces on the database storage paths. If there is no more space on any of the file systems on which the database storage paths are defined, automatic storage table spaces will be unable to increase in size and may become full.

The indicator is calculated using the formula:

$$(db.auto\_storage\_used / db.auto\_storage\_total) * 100$$

where

- *db.auto\_storage\_used* is the sum of used space across all physical file systems identified in the list of database storage paths
- *db.auto\_storage\_total* is the sum of total space across all physical file systems identified in the list of database storage paths

Database automatic storage path utilization is measured as a percentage of the space consumed on the database storage path file systems, where a high percentage indicates less than optimal function for this indicator.

The “Time to fullness” in the Additional Information line returned for this health indicator is a prediction of how much time is remaining until the maximum size for the table space has been reached.

**Usage note**

If you use storage groups, this health indicator indicates the consumption of storage for the defined database storage paths in the default storage group only.

*ts.ts\_auto\_resize\_status* - Table space automatic resize status health indicator:

This health indicator identifies whether table space resize operations are succeeding for DMS table spaces which have automatic resize enabled. When a DMS table space with automatic resize enabled fails to increase in size, it is

effectively full. This condition may be due to lack of free space on the file systems on which the table space containers are defined, or a result of the table space automatic resize settings. For example, the defined maximum size may have been reached, or the increase amount may be set too high to be accommodated by the remaining free space.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

ts.ts\_auto\_resize\_status

**Health monitor level**

Table Space

**Category**

Table Space Storage

**Type** State-based

**Unit** Not applicable

*ts.ts\_util\_auto\_resize - Automatic resize table space utilization health indicator:*

This health indicator tracks the consumption of storage for each automatic resize DMS table space on which a maximum size has been defined. The DMS table space is considered full when the maximum size has been reached.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

ts.ts\_util\_auto\_resize

**Health monitor level**

Table Space

**Category**

Table Space Storage

**Type** Upper-bounded threshold-based

**Unit** Percentage

The indicator is calculated using the formula:

$$((ts.used * ts.page_size) / ts.max_size) * 100$$

where

- *ts.used* is the value of “tablespace\_used\_pages - Used pages in table space monitor element” on page 1293
- *ts.page\_size* is the value of “tablespace\_page\_size - Table space page size monitor element” on page 1281

- *ts.max\_size* is the value of “tablespace\_max\_size - Maximum table space size” on page 1278

Automatic resize DMS table space utilization is measured as a percentage of the maximum table space storage consumed. A high percentage indicates the table space is approaching fullness. The short term and long term growth rates, included in the additional information for this indicator, can be used to determine if the current rate of growth is a short term aberration, or consistent with long term growth.

The “Time to fullness” in the Additional Information line returned for this health indicator is a prediction of how much time is remaining until the maximum size for the table space has been reached.

*ts.ts\_util* - Table Space Utilization:

This health indicator tracks the consumption of storage for each DMS table space.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

ts.ts\_util

**Health monitor level**

Table Space

**Category**

Table Space Storage

**Type** Upper-bounded threshold-based

**Unit** Percentage

The DMS table space is considered full when all containers are full.

If automatic resize is enabled on the table space, this health indicator will not be evaluated. Instead, the database automatic storage utilization **db.auto\_storage\_util** and table space automatic resize status (**ts.ts\_auto\_resize\_status**) health indicators are relevant for table space storage monitoring. The automatic resize table space utilization (**ts.ts\_util\_auto\_resize**) health indicator will also be available if a maximum size was defined on this table space. The table space utilization percentage can still be retrieved from column TBSP\_UTILIZATION\_PERCENT of the TBSP\_UTILIZATION administrative view if it is required.

The indicator is calculated using the formula:

$$(ts.used / ts.usable) * 100$$

where

- *ts.used* is the value of “tablespace\_used\_pages - Used pages in table space monitor element” on page 1293
- *ts.usable* is the value of “tablespace\_usable\_pages - Usable pages in table space monitor element” on page 1293

Table space utilization is measured as the percentage of space consumed, where a high percentage indicates less than optimal function for this indicator.

The short term and long term growth rates, included in the additional information for this indicator, can be used to determine if current rate of growth is a short term aberration or consistent with longer term growth.

The “Time to fullness” in the Additional Information line returned for this health indicator is a prediction of how much time is remaining until the maximum size for the table space has been reached.

*tsc.tscont\_util - Table Space Container Utilization:*

This health indicator tracks the consumption of storage for each SMS table space that is not using automatic storage.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

tsc.tscont\_util

**Health monitor level**

Table Space Container

**Category**

Table Space Storage

**Type** Upper-bounded threshold-based

**Unit** Percentage

An SMS table space is considered full if there is no more space on any of the file systems for which containers are defined.

If free space is not available on the file system to expand an SMS container, the associated table space becomes full.

An alert may be issued for each container defined on the file system that is running out of free space.

The indicator is calculated using the formula:

$(fs.used / fs.total) * 100$

where fs is the file system in which the container resides.

SMS table space utilization is measured as the percentage of space consumed, where a high percentage indicates less than optimal function for this indicator.

The short term and long term growth rates, included in the additional information for this indicator, can be used to determine if current rate of growth is a short term aberration or consistent with longer term growth.

The “Time to fullness” in the Additional Information line returned for this health indicator is a prediction of how much time is remaining until the maximum size for the table space has been reached.

*ts.ts\_op\_status - Table Space Operational State:*

The state of a table space can restrict activity or tasks that can be performed. A change from normal to another state may generate an Attention alert.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

ts.ts\_op\_status

**Health monitor level**

Table Space

**Category**

Table Space Storage

**Type** State-based

**Unit** Not applicable

*tsc.tscont\_op\_status - Table Space Container Operational State:*

This health indicator tracks the accessibility of the table space container. The accessibility of the container can restrict activity or tasks that can be performed. If the container is not accessible, an Attention alert may be generated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

tsc.tscont\_op\_status

**Health monitor level**

Table Space Container

**Category**

Table Space Storage

**Type** State-based

**Unit** Not applicable

**Sorting health indicators:**

*db2.sort\_privmem\_util - Private Sort Memory Utilization:*

This indicator tracks the utilization of the private sort memory. If `db2.sort_heap_allocated` (system monitor element)  $\geq$  `sheapthres` (DBM

configuration parameter), sorts may not be getting full sort heap as defined by the *sortheap* parameter and an alert may be generated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db2.sort\_privmem\_util

**Health monitor level**

Database

**Category**

Sorting

**Type** Upper-bounded threshold-based

**Unit** Percentage

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

The indicator is calculated using the formula:

$(db2.sort\_heap\_allocated / sheapthres) * 100$

The Post Threshold Sorts snapshot monitor element measures the number of sorts that have requested heaps after the sort heap threshold has been exceeded. The value of this indicator, shown in the Additional Details, indicates the degree of severity of the problem for this health indicator.

The Maximum Private Sort Memory Used snapshot monitor element maintains a private sort memory high watermark for the instance. The value of this indicator, shown in the Additional Information, indicates the maximum amount of private sort memory that has been in use at any one point in time since the instance was last recycled. This value can be used to help determine an appropriate value for *sheapthres*.

*db.sort\_shrmem\_util* - Shared Sort Memory Utilization:

This indicator tracks the utilization of the shared sort memory. The *sheapthres\_shr* database configuration parameter is a hard limit. If the allocation is close to the limit, an alert may be generated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.sort\_shrmem\_util

**Health monitor level**

Database

**Category**

Sorting

**Type** Upper-bounded threshold-based**Unit** Percentage

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

The indicator is calculated using the formula:

$$(db.sort\_shrheap\_allocated / sheapthres\_shr)*100$$

Note that if *sheapthres\_shr* is set to 0, then *sheapthres* serves as the shared sortheap threshold.

The Maximum Shared Sort Memory Used snapshot monitor element maintains a shared sort memory high watermark for the database. The value of this indicator, shown in the Additional Information, indicates the maximum amount of shared sort memory that has been in use at any one point in time since the database has been active. This value can be used to help determine an appropriate value for the shared sort memory threshold.

Consider using the self-tuning memory feature to have sort memory resources automatically allocated as required by the current workload. If you have the self tuning memory feature enabled for the sort memory area, you should configure this health indicator to disable threshold checking.

*db.spilled\_sorts* - Percentage of Sorts That Overflowed:

Sorts that overflow to disk can cause significant performance degradation. If this occurs, an alert may be generated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.spilled\_sorts

**Health monitor level**

Database

**Category**

Sorting

**Type** Upper-bounded threshold-based**Unit** Percentage

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

The indicator is calculated using the formula:

$$(db.sort\_overflows_t - db.sort\_overflows_{t-1}) / (db.total\_sorts_t - db.total\_sorts_{t-1}) * 100$$

where  $t$  is the current snapshot and  $t-1$  is a snapshot 1 hour ago. The system monitor element `db.sort_overflows` (based on the `sort_overflows` monitor element) is the total number of sorts that ran out of sort heap and may have required disk space for temporary storage. The element `db.total_sorts` (based on the `total_sorts` monitor element) is the total number of sorts that have been executed.

Consider using the self-tuning memory feature to have sort memory resources automatically allocated as required by the current workload. If you have the self tuning memory feature enabled for the sort memory area, you should configure this health indicator to disable threshold checking.

*db.max\_sort\_shrmem\_util - Long Term Shared Sort Memory Utilization:*

This indicator tracks an over-configured shared sort heap, looking to see if there are resources that can be freed for use somewhere else in the DB2 database system.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

`db.max_sort_shrmem_util`

**Health monitor level**

Database

**Category**

Sorting

**Type** Lower-bounded threshold-based

**Unit** Percentage

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and if sorts do not overflow unnecessarily.

An alert might be generated when the percentage usage is low.

The indicator is calculated using the formula:

$(db.max\_shr\_sort\_mem / sheaphres\_shr) * 100$

The system monitor element `db.max_shr_sort_mem` (based on the `sort_shrheap_top` monitor element) is the high watermark for shared sort memory usage.

Consider using the self-tuning memory feature to have sort memory resources automatically allocated as required by the current workload. If you have the self-tuning memory feature enabled for the sort memory area, you should configure this health indicator to disable threshold checking.

**Database manager (DBMS) health indicators:**

*db2.db2\_op\_status - Instance Operational State:*

An instance is considered healthy if the instance state does not restrict activity or tasks being performed.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db2.db2\_op\_status

**Health monitor level**

Instance

**Category**

DBMS

**Type** State-based

**Unit** Not applicable

The state can be one of the following values: Active, Quiesce pending, Quiesced, or Down. A non-Active state may generate an Attention alert.

The health monitor is unable to execute actions for the db2.db2\_op\_status health indicator if the indicator enters the down state. This state can arise, for example, when an instance that the indicator is monitoring becomes inactive because of an explicit stop request or an abnormal termination. If you want to have the instance restart automatically after any abnormal termination, you can configure the fault monitor (**db2fm**) to keep the instance highly available.

*Instance Highest Severity Alert State:*

This indicator represents the rolled-up alert state of an instance being monitored. The alert state of an instance is the highest alert state of the instance and its databases, and database objects being monitored.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

Not applicable. This health indicator does not have configuration or recommendations support.

**Health monitor level**

Instance

**Category**

DBMS

**Type** State-based

**Unit** Not applicable

The order of the alert states is as follows:

- Alarm
- Warning

- Attention
- Normal

The alert state of the instance determines the overall health of the DB2 database system.

**Database health indicators:**

*db.db\_op\_status - Database Operational State:*

The state of the database can restrict activity or tasks that can be performed. The state can be one of the following values: Active, Quiesce pending, Quiesced, or Rollforward. A change from Active to another state may generate an Attention alert.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.db\_op\_status

**Health monitor level**

Database

**Category**

Database

**Type** State-based

**Unit** Not applicable

*Database Highest Severity Alert State:*

This indicator represents the rolled-up alert state of the database being monitored. The alert state of a database is the highest alert state of the database and its objects.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

Not applicable. This health indicator does not have configuration or recommendations support.

**Health monitor level**

Database

**Category**

Database

**Type** State-based

**Unit** Not applicable

The order of the alert states is as follows:

- Alarm
- Warning
- Attention
- Normal

**Maintenance health indicators:**

*db.tb\_reorg\_req - Reorganization Required:*

This health indicator tracks the need to reorganize tables or indexes within a database. Tables or all indexes defined on a table require reorganization to eliminate fragmented data. The reorganization is accomplished by compacting the information and reconstructing the rows or index data. The result might yield an improved performance and freed space in the table or indexes.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.tb\_reorg\_req

**Health monitor level**

Database

**Category**

Database Maintenance

**Type** Collection state-based

**Unit** Not applicable

You can filter the set of tables evaluated by this health indicator by specifying in your automatic maintenance policy the names of the tables to be evaluated. This can be done using the Automatic Maintenance wizard.

An attention alert might be generated to indicate that reorganization is required. Reorganization can be automated by setting the AUTO\_REORG database configuration parameter to ON. If automatic reorganization is enabled, the attention alert indicates either that one or more automatic reorganizations could not complete successfully or that there are tables which require reorganization, but automatic reorganization is not being performed because the size of the table per database partition exceeds the maximum size criteria for tables that should be considered for offline reorganization. Refer to the collection details of this health indicator for the list of objects that need attention.

*db.tb\_runstats\_req - Statistics Collection Required:*

This health indicator tracks the need to collect statistics for tables and their indexes within a database. Tables and all indexes defined on a table require statistics to improve query execution time.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health

monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.tb\_runstats\_req

**Health monitor level**

Database

**Category**

Database Maintenance

**Type** Collection state-based

**Unit** Not applicable

The tables considered by this health indicator can be limited using an SQL query. The scope in the additional information displays the subselect clause on system tables for this query.

An attention alert may be generated to indicate that statistics collection is required. Statistics can be automatically collected by setting the AUTO\_RUNSTATS database configuration parameter to ON. If automatic statistics collection is enabled, the attention alert indicates that one or more automatic statistics collection actions did not complete successfully.

*db.db\_backup\_req - Database Backup Required:*

This health indicator tracks the need for a backup on the database. Backups should be taken regularly as part of a recovery strategy to protect your data against the possibility of loss in the event of a hardware or software failure.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.db\_backup\_req

**Health monitor level**

Database

**Category**

Database Maintenance

**Type** State-based

**Unit** Not applicable

This health indicator determines when a database backup is required based on the time elapsed and amount of data changed since the last backup.

An attention alert might be generated to indicate that a database backup is required. Database backups can be automated by setting the AUTO\_DB\_BACKUP

database configuration parameter to ON. If automatic database backups are enabled, the attention alert indicates that one or more automatic database backups did not complete successfully.

### High availability disaster recovery (HADR) health indicators:

*db.hadr\_op\_status* - HADR Operational Status:

This health indicator tracks the high availability disaster recovery (HADR) operational state of the database. The state between primary and standby servers can be one of the following values: Connected, Congested or Disconnected. A change from Connected to another state might generate an Attention alert.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### Identifier

db.hadr\_op\_status

#### Health monitor level

Database

#### Category

High availability disaster recovery

**Type** State-based

**Unit** Not applicable

*db.hadr\_delay* - HADR Log Delay:

This health indicator tracks the current average delay (in minutes) between the data changes on the primary database and the replication of those changes on the standby database. With a large delay value, data loss can occur when failing over to the standby database after a failure on the primary database. A large delay value can also mean longer downtime when takeover is required, because the primary database is ahead of the standby database.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### Identifier

db.hadr\_delay

#### Health monitor level

Database

#### Category

High availability disaster recovery

**Type** Upper-bounded threshold-based

**Unit** Minutes

## Logging health indicators:

### *db.log\_util - Log Utilization:*

This indicator tracks the total amount of active log space used in bytes in the database.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.log\_util

#### **Health monitor level**

Database

#### **Category**

Logging

**Type** Upper-bounded threshold-based

**Unit** Percentage

Log utilization is measured as the percentage of space consumed, where a high percentage may generate an alert.

The indicator is calculated using the formula:

$$(db.total\_log\_used / (db.total\_log\_used + db.total\_log\_available)) * 100$$

The values for the log-related database configuration parameters, shown in the additional information, display the current allocations for logs. The additional information also includes the application id for the application which has the oldest active transaction. This application can be forced to free up log space.

### *db.log\_fs\_util - Log File System Utilization:*

Log File System Utilization tracks the fullness of the file system on which the transaction logs reside.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.log\_fs\_util

#### **Health monitor level**

Database

#### **Category**

Logging

**Type** Upper-bounded threshold-based

**Unit** Percentage

The DB2 database system may not be able to create a new log file if there is no room on the file system.

Log utilization is measured as the percentage of space consumed. If the amount of free space in the file system is minimal (that is, there is a high percentage for utilization), an alert may be generated.

The indicator is calculated using the formula:  $(fs.log\_fs\_used / fs.log\_fs\_total) * 100$  where fs is the file system on which the log resides.

The values for the log-related database configuration parameters, shown in the additional information, display the current allocations for logs. The additional details also shows if user exit is enabled.

If Block on Log Disk Full, shown in the additional details, is set to yes and utilization is at 100%, you should resolve any alerts as soon as possible to limit the impact to applications which cannot commit transactions until the log file is successfully created.

#### **Application concurrency health indicators:**

*db.deadlock\_rate* - *Deadlock Rate*:

Deadlock rate tracks the rate at which deadlocks are occurring in the database and the degree to which applications are experiencing contention problems.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.deadlock\_rate

#### **Health monitor level**

Database

#### **Category**

Application Concurrency

**Type** Upper-bounded threshold-based

**Unit** Deadlocks per hour

Deadlocks may be caused by the following situations:

- Lock escalations are occurring for the database
- An application may be locking tables explicitly when system-generated row locks may be sufficient
- An application may be using an inappropriate isolation level when binding
- Catalog tables are locked for repeatable read
- Applications are getting the same locks in different orders, resulting in deadlock.

The indicator is calculated using the formula:

$(db.deadlocks_t - db.deadlocks_{t-1})$

where  $t$  is the current snapshot and  $t-1$  is the last snapshot, taken 60 minutes before the current snapshot.

The higher the rate of deadlocks, the greater the degree of contention which may generate an alert.

*db.locklist\_util - Lock List Utilization:*

This indicator tracks the amount of lock list memory that is being used.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.locklist\_util

**Health monitor level**

Database

**Category**

Application Concurrency

**Type** Upper-bounded threshold-based

**Unit** Percentage

There is one lock list per database and it contains the locks held by all applications concurrently connected to the database. There is a set limit on lock list memory. Once the limit is reached, performance degrades because of the following situations:

- Lock escalation converts row locks to table locks, thereby reducing concurrency on shared objects in the database.
- More deadlocks between applications can occur since applications are waiting for a limited number of table locks. As a result, transactions are rolled back.

An error is returned to the application when the maximum number of lock requests has reached the limit set for the database.

The indicator is calculated using the formula:

$(db.lock\_list\_in\_use / (locklist * 4096)) * 100$

Utilization is measured as a percentage of memory consumed, where a high percentage represents an unhealthy condition.

Consider using the self-tuning memory feature to have lock memory resources automatically allocated as required by the current workload. If you have the self-tuning memory feature enabled for the lock memory area, you should configure this health indicator to disable threshold checking.

*db.lock\_escal\_rate - Lock Escalation Rate:*

This indicator tracks the rate at which locks have been escalated from row locks to a table lock thereby impacting transaction concurrency.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.lock\_escal\_rate

**Health monitor level**

Database

**Category**

Application Concurrency

**Type** Upper-bounded threshold-based

**Unit** Lock escalations per hour

A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the *maxlocks* and *locklist* database configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, the application uses the space in the lock list allocated for other applications. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database. When the entire lock list is full, an error occurs.

The indicator is calculated using the formula:

$$(db.lock\_escal_s_t - db.lock\_escal_s_{t-1})$$

where 't' is the current snapshot and 't-1' is the last snapshot, taken 60 minutes before the current snapshot.

The higher the rate of deadlocks, the greater the degree of contention which may generate an alert.

Consider using the self-tuning memory feature to have lock memory resources automatically allocated as required by the current workload. If you have the self-tuning memory feature enabled for the lock memory area, you should configure this health indicator to disable threshold checking.

*db.apps\_waiting\_locks* - Percentage of Applications Waiting on Locks:

This indicator measures the percentage of all currently executing applications that are waiting on locks.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information,

see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.apps\_waiting\_locks

**Health monitor level**

Database

**Category**

Application Concurrency

**Type** Upper-bounded threshold-based

**Unit** Percentage

A high percentage can indicate that applications are experiencing concurrency problems which can negatively affect performance.

The indicator is calculated using the formula:

$(db.locks\_waiting / db.appls\_cur\_cons) * 100$

**Package cache, catalog cache, and workspace health indicators:**

*db.catcache\_hitratio - Catalog Cache Hit Ratio:*

The hit ratio is a percentage indicating how well the catalog cache is helping to avoid actual accesses to the catalog on disk. A high ratio indicates it is successful in avoiding actual disk I/O accesses.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.catcache\_hitratio

**Health monitor level**

Database

**Category**

Package and Catalog Caches, and Workspaces

**Type** Lower-bounded threshold-based

**Unit** Percentage

The indicator is calculated using the formula:

$(1 - (db.cat\_cache\_inserts/db.cat\_cache\_lookups)) * 100$

*db.pkgcache\_hitratio - Package Cache Hit Ratio:*

The hit ratio is a percentage indicating how well the package cache is helping to avoid reloading packages and sections for static SQL from the system catalogs as well as helping to avoid recompiling dynamic SQL statements. A high ratio indicates it is successful in avoiding these activities.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Identifier**

db.pkgcache\_hitratio

**Health monitor level**

Database

**Category**

Package and Catalog Caches, and Workspaces

**Type** Lower-bounded threshold-based

**Unit** Percentage

The indicator is calculated using the formula:

$(1 - (\text{db.pkg\_cache\_inserts} / \text{db.pkg\_cache\_lookups})) * 100$

Consider using the self-tuning memory feature to have package cache memory resources automatically allocated as required by the current workload. If you have the self-tuning memory feature enabled for the package cache memory area, you should configure this health indicator to disable threshold checking.

*db.shrworkspace\_hitratio - Shared Workspace Hit Ratio:*

The hit ratio is a percentage indicating how well the shared SQL workspace is helping to avoid having to initialize sections for SQL statements that are about to be executed. A high ratio indicates it is successful in avoiding this action.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

**Note:** The **db.shrworkspace\_hitratio** health indicator is deprecated starting with DB2 Version 9.5. Using this health indicator will not generate an error. However, it does not return a valid value. This indicator is no longer recommended and might be removed in a future release.

**Identifier**

db.shrworkspace\_hitratio

**Health monitor level**

Database

**Category**

Package and Catalog Caches, and Workspaces

**Type** Lower-bounded threshold-based

**Unit** Percentage

The indicator is calculated using the formula:

$(1 - (\text{db.shr\_workspace\_section\_inserts} / \text{db.shr\_workspace\_section\_lookups})) * 100$

## Memory health indicators:

### *db2.mon\_heap\_util - Monitor Heap Utilization:*

This indicator tracks the consumption of the monitor heap memory, based on the memory pool with the ID SQLM\_HEAP\_MONITOR.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db2.mon\_heap\_util

#### **Health monitor level**

Instance

#### **Category**

Memory

**Type** Upper-bounded threshold-based

**Unit** Percentage

The utilization is calculated using the formula:

$(db2.pool\_cur\_size / db2.pool\_config\_size) * 100$

for the Memory Pool Identifier SQLM\_HEAP\_MONITOR.

Once this percentage reaches the maximum, 100%, monitor operations may fail.

### *db.db\_heap\_util - Database Heap Utilization:*

This indicator tracks the consumption of the monitor heap memory, based on the memory pool with the ID SQLM\_HEAP\_DATABASE.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.db\_heap\_util

#### **Health monitor level**

Database

#### **Category**

Memory

**Type** Upper-bounded threshold-based

**Unit** Percentage

The utilization is calculated using the formula

$(db.pool\_cur\_size / db.pool\_config\_size) * 100$

for the Memory Pool Identifier SQLM\_HEAP\_DATABASE.

Once this percentage reaches the maximum, 100%, queries and operations may fail because there is no heap available.

#### **Federated health indicators:**

*db.fed\_nicknames\_op\_status - Nickname Status:*

This health indicator checks all of the nicknames defined in a federated database to determine if there are any invalid nicknames. A nickname may be invalid if the data source object was dropped or changed, or if the user mapping is incorrect.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.fed\_nicknames\_op\_status

#### **Health monitor level**

Database

#### **Category**

Federated

**Type** Collection state-based

**Unit** Not applicable

An attention alert might be generated if any nicknames defined in the federated database are invalid. Refer to the collection details of this health indicator for the list of objects that need attention.

The FEDERATED database manager parameter must be set to YES for this health indicator to check nicknames status.

*db.fed\_servers\_op\_status - Data Source Server Status:*

This health indicator checks all of the data source servers defined in a federated database to determine if any are unavailable. A data source server might be unavailable if the data source server was stopped, no longer exists, or was incorrectly configured.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### **Identifier**

db.fed\_servers\_op\_status

#### **Health monitor level**

Database

**Category**

Federated

**Type** Collection state-based**Unit** Not applicable

An attention alert might be generated if any nicknames defined in the federated database are not valid. Refer to the collection details of this health indicator for the list of objects that need attention.

The FEDERATED database manager parameter must be set to YES for this health indicator to check data source server status.

**Health monitor interfaces**

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

The following table lists the health monitor interfaces for APIs:

*Table 134. Health monitor interfaces: APIs*

| Monitoring task  | API  |
|--|--|
| Capturing a health snapshot  | db2GetSnapshot - Get Snapshot with snapshot class SQLM_CLASS_HEALTH  |
| Capturing a health snapshot with the full list of collection objects   | db2GetSnapshot - Get Snapshot with snapshot class SQLM_CLASS_HEALTH and SQLM_HMON_OPT_COLL_FULL for agent_id             |
| Capturing a health snapshot with formula, additional information, and history                                      | db2GetSnapshot - Get Snapshot with snapshot class SQLM_CLASS_HEALTH_WITH_DETAIL  |
| Capturing a health snapshot with formula, additional information, history, and the full list of collection objects | db2GetSnapshot - Get Snapshot with snapshot class SQLM_CLASS_HEALTH_WITH_DETAIL and SQLM_HMON_OPT_COLL_FULL for agent_id |
| Converting the self-describing data stream   | db2ConvMonStream - Convert Monitor stream  |
| Estimating the size of a health snapshot   | db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer   |

The **GET HEALTH SNAPSHOT** command is part of the health monitor components that have been deprecated.

The following table lists the health monitor interfaces for CLP commands:

*Table 135. Health monitor interfaces: CLP commands*

| Monitoring task   | CLP command                              |
|---|--|
| Capturing a health snapshot   | GET HEALTH SNAPSHOT Command              |
| Capturing a health snapshot with formula, additional information, and history | GET HEALTH SNAPSHOT WITH DETAILS Command |

The health monitor SQL functions are part of the health monitor components that have been deprecated.

The following table lists the health monitor interfaces for SQL functions:

*Table 136. Health monitor interfaces: SQL functions*

| Monitoring task   | SQL Function       |
|---|--------------------|
| Database manager level health information snapshot            | HEALTH_DBM_INFO    |
| Database manager level health indicator snapshot              | HEALTH_DBM_HI      |
| Database manager level health indicator history snapshot      | HEALTH_DBM_HI_HIS  |
| Database level health information snapshot                    | HEALTH_DB_INFO     |
| Database level health indicator snapshot                      | HEALTH_DB_HI       |
| Database level health indicator history snapshot              | HEALTH_DB_HI_HIS   |
| Database level health indicator collection snapshot           | HEALTH_DB_HIC      |
| Database level health indicator collection history snapshot   | HEALTH_DB_HIC_HIS  |
| Table space level health information snapshot                 | HEALTH_TBS_INFO    |
| Table space level health indicator snapshot                   | HEALTH_TBS_HI      |
| Table space level health indicator history snapshot           | HEALTH_TBS_HI_HIS  |
| Table space container level health information snapshot       | HEALTH_CONT_INFO   |
| Table space container level health indicator snapshot         | HEALTH_CONT_HI     |
| Table space container level health indicator history snapshot | HEALTH_CONT_HI_HIS |

#### Health monitor SQL table functions:

The health monitor SQL table functions have been deprecated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

The following table lists all of the snapshot table functions. Each table function corresponds to a health snapshot request type.

*Table 137. Snapshot monitor SQL table functions*

| Monitor level    | SQL table function | Information returned  |
|------------------|--------------------|---|
| Database manager | HEALTH_DBM_INFO    | Basic information about the health snapshot from the database manager level |
| Database manager | HEALTH_DBM_HI      | Health indicator information from the database manager level                |
| Database manager | HEALTH_DBM_HI_HIS  | Health indicator history information from the database manager level        |
| Database         | HEALTH_DB_INFO     | Basic information about the health snapshot from a database                 |
| Database         | HEALTH_DB_HI       | Health indicator information from a database                                |
| Database         | HEALTH_DB_HI_HIS   | Health indicator history information from a database                        |

Table 137. Snapshot monitor SQL table functions (continued)

| Monitor level | SQL table function | Information returned  |
|---------------|--------------------|---|
| Database      | HEALTH_DB_HIC      | Collection information for collection health indicators for a database          |
| Database      | HEALTH_DB_HIC_HIS  | Collection history information for collection health indicators for a database  |
| Table space   | HEALTH_TBS_INFO    | Basic information about the health snapshot for the table spaces for a database |
| Table space   | HEALTH_TBS_HI      | Health indicator information about the table spaces for a database              |
| Table space   | HEALTH_TBS_HI_HIS  | Health indicator history information about the table spaces for a database      |
| Table space   | HEALTH_CONT_INFO   | Basic information about the health snapshot for the containers for a database   |
| Table space   | HEALTH_CONT_HI     | Health indicator information about the containers for a database                |
| Table space   | HEALTH_CONT_HI_HIS | Health indicator history information about the containers for a database        |

#### Health monitor CLP commands:

You can get health status information for the database manager and its databases by issuing health monitor commands.

The information returned represents a snapshot of the health state at the time the command was issued.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

The following table lists all the supported snapshot request types.

Table 138. Snapshot monitor CLP commands

| Monitor level    | CLP command   | Information returned   |
|------------------|---|--|
| Database manager | get health snapshot for dbm                               | Database manager level information.  |
| Database         | get health snapshot for all databases                     | Database level information. Information is returned only if the database is activated.                                   |
| Database         | get health snapshot for database on <i>database-alias</i> | Database level information. Information is returned only if the database is activated.                                   |
| Database         | get health snapshot for all on <i>database-alias</i>      | Database, table space, and table space container information. Information is returned only if the database is activated. |

Table 138. Snapshot monitor CLP commands (continued)

| Monitor level | CLP command   | Information returned  |
|---------------|---|---|
| Table space   | get snapshot for tablespaces on <i>database-alias</i> | Table space level information for each table space that was accessed by an application connected to the database. Also includes health information for each table space container within the table space. |

### Health monitor API request types:

The following table lists all the supported snapshot request types.

Table 139. Snapshot Monitor API Request Types

| Monitor level    | API request type        | Information returned   |
|------------------|-------------------------|--|
| Database manager | SQLMA_DB2               | Database manager level information.  |
| Database         | SQLMA_DBASE_ALL         | Database level information. Information is returned only if the database is activated.   |
| Database         | SQLMA_DBASE             | Database level information. Information is returned only if the database is activated.   |
| Table space      | SQLMA_DBASE_TABLESPACES | Table space level information for each table space that has been accessed by an application connected to the database. Also includes health information for each table space container within the table space. |

### Health monitor interface mappings to logical data groups

The following table lists all the supported health snapshot request types.

Table 140. Health monitor interface mappings to logical data groups

| API request type   | CLP command  | SQL table function | Logical data groups                   |
|--|--|--------------------|---------------------------------------|
| SQLMA_DB2  | get health snapshot for dbm  | HEALTH_DBM_INFO    | db2                                   |
|  |  | HEALTH_DBM_HI      | health_indicator                      |
| SQLMA_DBASE  | get health snapshot for database on <i>dbname</i>                      | HEALTH_DB_INFO     | dbase                                 |
|  |  | HEALTH_DB_HI       | health_indicator                      |
|  |  | HEALTH_DB_HI_HIS   | health_indicator_history              |
| SQLMA_DBASE with SQLM_HMON_OPT_COLL_FULL in the agent_id | get health snapshot for database on <i>dbname</i> with full collection | HEALTH_DB_HIC      | health_indicator, hi_obj_list         |
|  |  | HEALTH_DB_HIC_HIST | health_indicator_history, hi_obj_list |
| SQLMA_DBASE_ALL  | get health snapshot for all databases                                  | HEALTH_DB_INFO     | dbase                                 |
|  |  | HEALTH_DB_HI       | health_indicator                      |
|  | get health snapshot for all databases show detail                      | HEALTH_DB_HI_HIS   | health_indicator_history              |

Table 140. Health monitor interface mappings to logical data groups (continued)

| API request type        | CLP command  | SQL table function | Logical data groups      |
|-------------------------|--|--------------------|--------------------------|
| SQLMA_DBASE_TABLESPACES | get health snapshot for tablespaces on <i>dbname</i>             | HEALTH_TS_INFO     | tablespace               |
|                         |  | HEALTH_TS_HI       | health_indicator         |
|                         |  | HEALTH_CONT_INFO   | tablespace_container     |
|                         |  | HEALTH_CONT_HI     | health_indicator         |
|                         | get health snapshot for tablespaces on <i>dbname</i> show detail | HEALTH_TS_HI_HIS   | health_indicator_history |
|                         |  | HEALTH_CONT_HI_HIS | health_indicator_history |

The following figure shows the order that logical data groupings can appear in a health snapshot data stream.

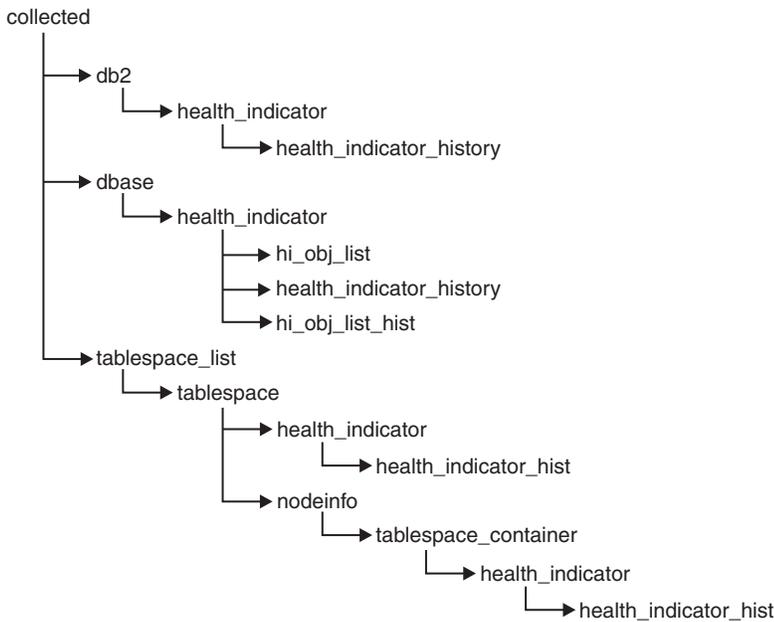


Figure 9. Health snapshot logical data groupings

## Enabling health alert notification

To enable email or pager notification when an alert is generated, you must set configuration parameters and specify contact information.

### Before you begin

The DB2 Administration Server (DAS) must be running on the system where the contact list is located. For example, if the CONTACT\_HOST configuration parameter is set to a remote system, the DAS must be running on the remote system in order for contacts to be notified of alerts.

### About this task

To enable health alert notification:

## Procedure

1. Specify the SMTP\_SERVER parameter. The DAS configuration parameter, SMTP\_SERVER, specifies the location of the mail server to use when sending both email and pager notification messages. Omit this step if the system where the DB2 database is installed is enabled as an unauthenticated SMTP server.
2. Specify the CONTACT\_HOST parameter. The DAS configuration parameter, CONTACT\_HOST, specifies the remote location of the contact list for all instances on the local system. By setting this parameter, a single contact list can be shared between multiple systems. Omit this step if you want to keep the contact list on the local system where the DB2 database is installed.
3. Specify the default contact for health monitor notification. To enable email or pager notification from the health monitor when an alert is generated, a default administration contact must be specified. If you choose not to provide this information, notification messages cannot be sent for alert conditions. You can provide the default administration contact information during installation, or you can defer the task until after installation is complete. If you choose to defer the task or want to add more contacts or groups to the notification list, you can specify contacts through the CLP, or C APIs:

•

### To specify contacts using the CLP:

To define an email contact as the default for health monitor notification, issue the following commands:

```
DB2 ADD CONTACT contact_name TYPE EMAIL ADDRESS  
      email_address DESCRIPTION 'Default Contact'
```

```
DB2 UPDATE NOTIFICATION LIST ADD CONTACT contact_name
```

For complete syntax details, see the Command Reference.

•

### To specify contacts using C APIs:

The following C code excerpt illustrates how to define health notification contacts:

```
...  
#include <db2ApiDf.h>  
  
SQL_API_RC rc = 0;  
struct db2AddContactData addContactData;  
struct sqlca sqlca;  
  
char* userid = "myuser";  
char* password = "pwd";  
char* contact = "DBA1";  
char* email = "dba1@mail.com";  
char* desc = "Default contact";  
  
memset(&addContactData, '\\0', sizeof(addContactData));  
memset (&sqlca, '\\0', sizeof(struct sqlca));  
  
addContactData.piUserId = userid;  
addContactData.piPassword = password;  
addContactData.piName = contact;  
addContactData.iType = DB2CONTACT_EMAIL;  
addContactData.piAddress = email;  
addContactData.iMaxPageLength = 0;  
addContactData.piDescription = desc;  
  
rc = db2AddContact(db2Version810, &addContactData, &sqlca);
```

```

if (rc == 0) {
    db2HealthNotificationListUpdate update;
    db2UpdateHealthNotificationListData data;
    db2ContactTypeData contact;

    contact.pName = contact;
    contact.contactType = DB2CONTACT_EMAIL;

    update.iUpdateType = DB2HEALTHNOTIFICATIONLIST_ADD;
    update.piContact = &contact;

    data.iNumUpdates = 1;
    data.piUpdates = &update;

    rc = db2UpdateHealthNotificationList (db2Version810, &data, &ca);
}
...

```

## Health monitor

The health monitor captures information about the database manager, database, table space, and table space containers.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

The health monitor calculates health indicators based on data retrieved from database system monitor elements, the operating system, and DB2 database. The health monitor can only evaluate health indicators on a database and its objects when the database is active. You can keep the database active either by starting it with the **ACTIVATE DATABASE** command or by maintaining a permanent connection to the database.

The health monitor retains a maximum of 10 history records for each health indicator. This history is stored in the *instance\_path*\hmonCache directory and is removed when the health monitor is stopped. The health monitor automatically prunes obsolete history records when the maximum number of records is reached.

Health monitor data is accessible through health snapshots. Each health snapshot reports the status for each health indicator based on its most recent refresh interval. The snapshots are useful for detecting existing database health problems and predicting potential poor health of the database environment. You can capture a health snapshot from the CLP, by using APIs in a C or C++ application, or by using the graphical administration tools.

Health monitoring requires an instance attachment. If an attachment to an instance was not established using the **ATTACH TO** command, then a default instance attachment to the local instance is created.

In partitioned database environments, snapshots can be taken at any partition of the instance, or globally using a single instance connection. A global snapshot aggregates the data collected at each partition and returns a single set of values.

## Usage notes

The health monitor is supported on all editions of the DB2 database.

On Windows, the service for the DB2 instance needs to run under an account with SYSADM authority. You can use the **-u** option on the **db2icrt** command, or use the Services folder on Windows and edit the Log On properties to use an account with administrator privilege.

The health monitor process runs as a DB2 fenced mode process. These processes appear as DB2FMP on Windows. On other platforms, the health monitor process appears as DB2ACD.

The DB2 Administration server must be running on the system where the health monitor resides for notifications to be sent and alert actions to be run. If remote scripts, tasks, or contact lists are used, the DB2 Administration server on the remote system must also be started.

The tools catalog database is required only for creating tasks. If you do not use alert task actions for any health indicator, the tools catalog database is not required by the health monitor.

### Health indicator data

The health monitor records a set of data for each health indicator on each database partition, including:

- Health indicator name
- Value
- Evaluation timestamp
- Alert state
- Formula, if applicable
- Additional information, if applicable
- History of up to ten of the most recent health indicator evaluations. Each history entry captures the following health indicator evaluations leading up to the current health indicator output:
  - Value
  - Formula (if applicable)
  - Alert state
  - Timestamp

The health monitor also tracks the highest severity alert state at the instance, database, and table space levels. At each level, this health indicator represents the highest severity alert existing for health indicators at that level, or any of the levels below it. For example, the highest severity alert state for an instance includes health indicators on the instance, any of its database, and any of the table spaces and table space containers for each of the databases.

### Capturing database health snapshots

#### Capturing a database health snapshot using SQL table functions:

You can capture database health snapshots using SQL table functions. Each available health snapshot table function corresponds to a health snapshot request type.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

### About this task

To capture a database health snapshots using SQL table functions:

#### Procedure

1. Identify the SQL table function you plan to use.

SQL table functions have two input parameters:

- A VARCHAR(255) for the database name
- An INT for the partition number (a value between 0 and 999). Enter the integer corresponding to the partition number you want to monitor. To capture a snapshot for the currently connected partition, enter a value of -1. To capture a global snapshot, enter a value of -2.

**Note:** The database manager snapshot SQL table functions are the only exception to this rule because they have only one parameter. The single parameter is for partition number. If you enter NULL for the database name parameter, the monitor uses the database defined by the connection through which the table function has been called.

2. Issue the SQL statement.

The following example captures a basic health snapshot for the currently connected partition, and on the database defined by the connection from which this table function call is made:

```
SELECT * FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1))
      as HEALTH_DB_INFO
```

You can also select individual monitor elements from the returned table. Each column in the returned table corresponds to a monitor element. Accordingly, the monitor element column names correspond directly to the monitor element names. The following statement returns only the db path and server platform monitor elements:

```
SELECT db_path, server_platform
      FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1 ) )
      as HEALTH_DB_INFO
```

### Capturing a database health snapshot using the CLP:

You can capture health snapshots using the GET HEALTH SNAPSHOT command from the CLP. The command syntax supports retrieval of health snapshot information for the different object types monitored by the health monitor.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

## Before you begin

You must have an instance attachment to capture a health snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To obtain a snapshot of a remote instance, you must first attach to that instance.

## About this task

To capture a database health snapshot using the CLP

### Procedure

1. From the CLP, issue the GET HEALTH SNAPSHOT command with the required parameters.  
In the following example, a database manager level health snapshot is captured immediately after starting the database manager.  

```
db2 get health snapshot for dbm
```
2. For partitioned database systems, you can capture a database snapshot specifically for a certain partition or globally for all partitions. To capture a health snapshot for a database on a specific partition (for example, partition number 2), issue the following command:  

```
db2 get health snapshot for db on sample at dbpartitionnum 2
```

To capture a database snapshot for all applications on all partitions, issue the following command:

```
db2 get health snapshot for db on sample global
```

The following command captures a health snapshot with additional detail, including the formula, additional information, and health indicator history:

```
db2 get health snapshot for db on sample show detail
```
3. For collection state-based health indicators, you can capture a database snapshot for all collection objects, regardless of state. The regular GET HEALTH SNAPSHOT FOR DB command returns all collection objects requiring an alert for all collection state-based health indicators.  
To capture a health snapshot for a database with all collection objects listed, issue the following command:  

```
db2 get health snapshot for db on sample with full collection
```

### Capturing a database health snapshot from a client application:

You can capture health snapshots using the snapshot monitor API in a C or C++ application. A number of different health snapshot request types can be accessed by specifying parameters in the db2GetSnapshot API.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

## Before you begin

You must be attached to an instance to capture a health snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To obtain a snapshot of a remote instance, you must first attach to that instance.

## Procedure

1. Include the `sqlmon.h` and `db2ApiDf.h` DB2 libraries in your code. These libraries are found in the `sqllib\include` directory.

```
#include <db2ApiDf.h>
#include <sqlmon.h>
```

2. Set the snapshot buffer unit size to 50 KB.

```
#define SNAPSHOT_BUFFER_UNIT_SZ 51200
```

3. Declare the `sqlma`, `sqlca`, `sqlm_collected`, and `db2GetSnapshotData` structures.

```
struct sqlma *pRequestedDataGroups;
struct sqlca sqlca;
memset (&sqlca, '\0', sizeof(struct sqlca));
struct sqlm_collected collectedData;
memset (&sqlm_collected, '\0', sizeof(struct sqlm_collected));
db2GetSnapshotData getSnapshotParam;
memset(&db2GetSnapshotData, '\0', sizeof(db2GetSnapshotData));
```

4. Initialize a pointer to contain the snapshot buffer, and to establish the buffer's size.

```
static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
sqluint32 outputFormat;
char *snapshotBuffer;
```

5. Initialize the `sqlma` structure and specify that the snapshot you are capturing is of database manager level information.

```
pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
memset(&pRequestedDataGroups, '\0', sizeof(struct pRequestedDataGroups));
pRequestedDataGroups->obj_num = 1;
pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
```

6. Initialize the buffer, which will hold the snapshot output.

```
snapshotBuffer = (char *)malloc(snapshotBufferSize);
memset (&snapshotBuffer, '\0', sizeof(snapshotBuffer));
```

7. Populate the `db2GetSnapshotData` structure with the snapshot request type (from the `sqlma` structure), buffer information, and other information required to capture a snapshot.

```
getSnapshotParam.piSqlmaData = pRequestedDataGroups;
getSnapshotParam.poCollectedData = &collectedData;
getSnapshotParam.poBuffer = snapshotBuffer;
getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_HEALTH;
```

8. Capture the health snapshot. Pass the following parameters:

- `db2GetSnapshotData` structure, which contains the information necessary to capture a snapshot
- A reference to the buffer where snapshot output is directed.

```
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
```

9. Include logic to handle buffer overflow. After a snapshot is taken, the `sqlcode` is checked for a buffer overflow. If a buffer overflow occurs, the buffer is cleared, reinitialized, and the snapshot is taken again.

```
while (sqlca.sqlcode == 1606)
{
    free(snapshotBuffer);
    snapshotBufferSize += SNAPSHOT_BUFFER_UNIT_SZ;
    snapshotBuffer = (char *)malloc(snapshotBufferSize);
}
```

```

if (snapshotBuffer == NULL)
{
    printf("\nMemory allocation error.\n");
    return;
}

getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.poBuffer = snapshotBuffer;
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}

```

10. Process the snapshot monitor data stream. Refer to the figure following these steps to see the snapshot monitor data stream.

11. Clear the buffer.

```

free(snapshotBuffer);
free(pRequestedDataGroups);

```

**Results**

After you capture a health snapshot with the db2GetSnapshot API, the API returns the health snapshot output as a self-describing data stream. The following example shows the data stream structure:

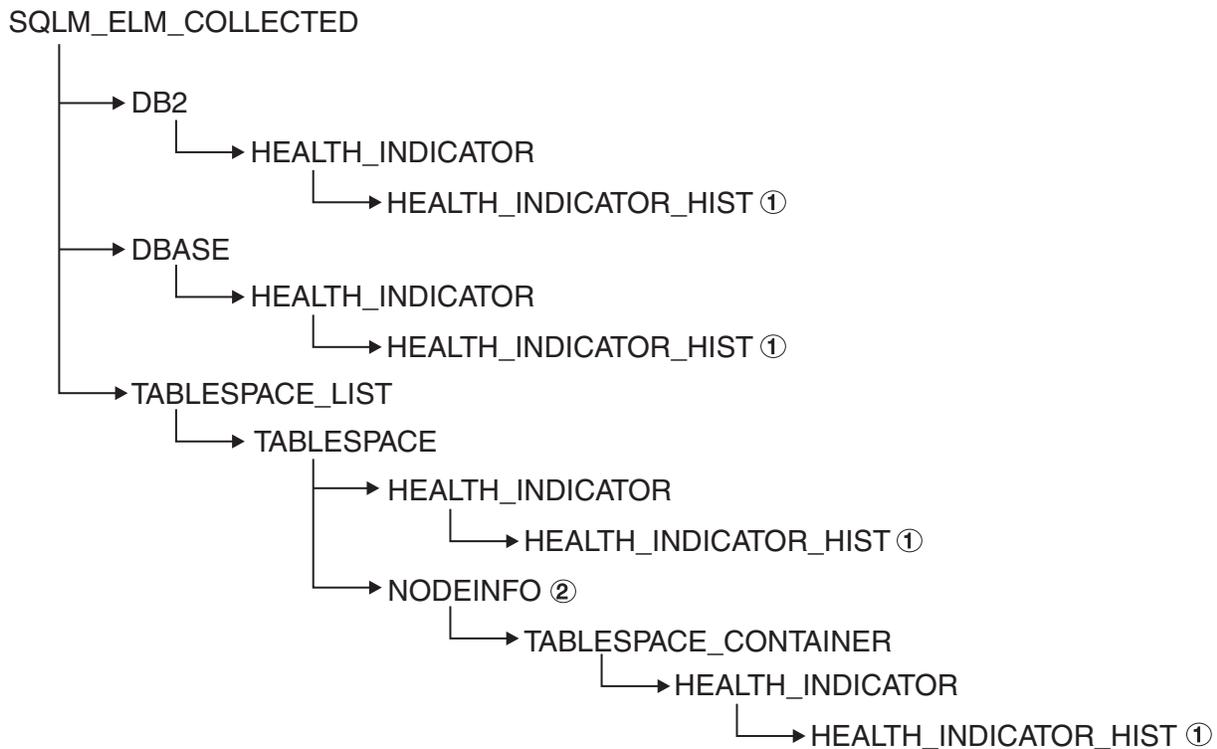


Figure 10. Health snapshot self-describing data stream

**Legend:**

1. Only available when the SQLM\_CLASS\_HEALTH\_WITH\_DETAIL snapshot class is used.
2. Only available in DB2 Enterprise Server Edition. Otherwise, table space container stream follows.

The following hierarchies display the specific elements in the health snapshot self-describing data stream.

The hierarchy of elements under SQLM\_ELM\_HI:

```
SQLM_ELM_HI
  SQLM_ELM_HI_ID
  SQLM_ELM_HI_VALUE
    SQLM_ELM_HI_TIMESTAMP
      SQLM_ELM_SECONDS
      SQLM_ELM_MICROSEC
  SQLM_ELM_HI_ALERT_STATE
```

The hierarchy of elements under SQLM\_ELM\_HI\_HIST, available only with the SQLM\_CLASS\_HEALTH\_WITH\_DETAIL snapshot class:

```
SQLM_ELM_HI_HIST
  SQLM_ELM_HI_FORMULA
  SQLM_ELM_HI_ADDITIONAL_INFO
  SQLM_ELM_HEALTH_INDICATOR_HIST
    SQLM_ELM_HI_ID
    SQLM_ELM_HI_VALUE
      SQLM_ELM_HI_TIMESTAMP
        SQLM_ELM_SECONDS
        SQLM_ELM_MICROSEC
    SQLM_ELM_HI_ALERT_STATE
  SQLM_ELM_HI_FORMULA
  SQLM_ELM_HI_ADDITIONAL_INFO
```

The hierarchy of elements under SQLM\_ELM\_OBJ\_LIST:

```
SQLM_ELM_HI_OBJ_LIST
  SQLM_ELM_HI_OBJ_NAME
  SQLM_ELM_HI_OBJ_DETAIL
  SQLM_ELM_HI_OBJ_STATE
  SQLM_ELM_HI_TIMESTAMP
    SQLM_ELM_SECONDS
    SQLM_ELM_MICROSEC
```

The hierarchy of elements under SQLM\_ELM\_OBJ\_LIST\_HIST, available only with the SQLM\_CLASS\_HEALTH\_WITH\_DETAIL snapshot class:

```
SQLM_ELM_HI_OBJ_LIST_HIST
  SQLM_ELM_HI_OBJ_NAME
  SQLM_ELM_HI_OBJ_STATE
  SQLM_ELM_HI_TIMESTAMP
    SQLM_ELM_SECONDS
    SQLM_ELM_MICROSEC
```

## Health monitor sample output

The following examples show health snapshots taken using the CLP, and their corresponding output, and illustrate the nature of the health monitor. The objective in the examples is to check the overall health status immediately after starting the database manager.

1. Take the database manager snapshot, using the GET HEALTH SNAPSHOT command:

```
db2 get health snapshot for dbm
```

After the GET HEALTH SNAPSHOT command is issued from the CLP, the snapshot output is directed to the screen.

```
Node name =
Node type = Database Server with local
           and remote clients
Instance name = DB2
Snapshot timestamp = 11-07-2002 12:43:23.613425

Number of database partitions in DB2 instance = 1
Start Database Manager timestamp = 11-07-2002 12:43:18.000108
```

Instance highest severity alert state = Not yet evaluated

Health Indicators:

Not yet evaluated

2. Analyze the output. From this health snapshot, you can see that the instance highest severity alert state is "Not yet evaluated". The instance is in this state because the health monitor has just started and has not yet evaluated any health indicators.

Should the instance highest severity alert state not change:

- Check the value of the HEALTH\_MON database manager configuration parameter to determine if the health monitor is on.
- If HEALTH\_MON=OFF, then the health monitor is not started. To start the health monitor, issue the UPDATE DBM CFG USING HEALTH\_MON ON command.
- If HEALTH\_MON=ON, attach to the instance to activate the health monitor. If an instance attachment exists, it is possible that the health monitor could not be loaded into memory.

Another example of taking a database health snapshot using the CLP is outlined below.

1. Before you begin, ensure that a database connection exists, and that the database is quiesced.
2. Take the database manager snapshot, using the GET HEALTH SNAPSHOT command:

```
db2 get health snapshot for db on sample
```

3. After the GET HEALTH SNAPSHOT command is issued from the CLP, the snapshot output is directed to the screen.

Database Health Snapshot

Snapshot timestamp = 12-09-2002 11:44:37.793184

Database name = SAMPLE  
Database path = E:\DB2\NODE0000\SQL00002\  
Input database alias = SAMPLE  
Operating system running at database server= NT  
Location of the database = Local  
Database highest severity alert state = Attention

Health Indicators:

```
...
Indicator Name = db.log_util
Value = 60
Unit = %
Evaluation timestamp = 12-09-2002 11:44:00.095000
Alert state = Normal

Indicator Name = db.db_op_status
Value = 2
Evaluation timestamp = 12-09-2002 11:44:00.095000
Alert state = Attention
```

4. Analyze the output.

This health snapshot reveals that there is an attention alert on the *db.db\_op\_status* health indicator. The value of 2 indicates that the database is in quiesced state.

## Global health snapshots

On a partitioned database system you can take a health snapshot of the current partition, a specified partition, or all partitions. When taking a global health snapshot across all the partitions of a partitioned database, data is aggregated, where possible, before the results are returned.

The aggregated alert state for the health indicator is equivalent to the highest severity alert state across all the database partitions. Additional information and history data cannot be aggregated across the database partitions, and therefore are not available. The remaining data for the health indicator is aggregated as detailed in the table below.

*Table 141. Aggregation of health indicator value, timestamp, and formula data*

| Health indicator   | Aggregation details  |
|--|--|
| <ul style="list-style-type: none"> <li>• db2.db2_op_status</li> <li>• db2.sort_privmem_util</li> <li>• db2.mon_heap_util</li> <li>• db.db_op_status</li> <li>• db.sort_shrmem_util</li> <li>• db.spilled_sorts</li> <li>• db.log_util</li> <li>• db.log_fs_util</li> <li>• db.locklist_util</li> <li>• db.apps_waiting_locks</li> <li>• db.db_heap_util</li> <li>• db.db_backup_req</li> <li>• ts.ts_util</li> </ul> | <p>The health indicator value is obtained from the partition that contains the highest value.</p> <p>The evaluation timestamp and formula are obtained from the same partition.</p>    |
| <ul style="list-style-type: none"> <li>• db.max_sort_shrmem_util</li> <li>• db.pkgcache_hitratio</li> <li>• db.catcache_hitratio</li> <li>• db.shrworkspace_hitratio</li> </ul>  | <p>The health indicator value is obtained from the partition that contains the lowest value.</p> <p>The evaluation timestamp and formula are obtained from the same partition.</p>     |
| <ul style="list-style-type: none"> <li>• db.deadlock_rate</li> <li>• db.lock_escal_rate</li> </ul>   | <p>The health indicator value is the sum of the values across all the database partitions.</p> <p>The evaluation timestamp and formula cannot be aggregated and are not available.</p> |
| <ul style="list-style-type: none"> <li>• ts.ts_op_status</li> <li>• tsc.tscont_op_status</li> <li>• tsc.tscont_util</li> </ul>   | <p>These health indicators is not aggregated.</p>  |
| <ul style="list-style-type: none"> <li>• db.hadr_op_status</li> <li>• db.hadr_log_delay</li> </ul>   | <p>These health indicators are not supported in a multiple partition database.</p>   |
| <ul style="list-style-type: none"> <li>• db.tb_reorg_req</li> <li>• db.tb_runstats_req</li> <li>• db.fed_nicknames_op_status</li> <li>• db.fed_servers_op_status</li> </ul>  | <p>This health indicator is evaluated only on one partition, so no aggregation is required. The data is returned from the partition which is evaluating the health indicator.</p>      |

**Note:** When taking a global snapshot on a single partition object, the output includes all the attributes because there are no partitions to aggregate.

## Retrieving health recommendations

### Health recommendation queries with SQL:

Recommendations can be queried with SQL using the SYSPROC.HEALTH\_HI\_REC stored procedure.

When using the SYSPROC.HEALTH\_HI\_REC stored procedure, recommendations are returned in an XML document that is:

- Formatted according to the health recommendations XML schema DB2RecommendationSchema.xsd located in the sql11ib\misc directory.
- Encoded in UTF-8 and contains text in the client language.
- Organized as a collection of recommendation sets, where each recommendation set describes a problem (health indicator) being resolved and contains one or more recommendations to resolve that health indicator. Refer to the schema definition for specific details about information that can be retrieved from the document.

All information available through the CLP is also available in the XML recommendation document that is returned when you query with SQL.

The SYSPROC.HEALTH\_HI\_REC stored procedure takes the following arguments:

- A health indicator
- A definition of the object on which the health indicator has entered an alert state

The output recommendation document is returned as a BLOB. Therefore, it is not helpful to work with this stored procedure from the command line, since the CLP will limit the amount of output displayed. It is recommended that this stored procedure be invoked using a high level language (such as C or Java) that allows the returned XML document to be properly parsed to retrieve any required elements and attributes.

### Retrieving health recommendations using the command line processor:

Recommendations can be retrieved using the GET RECOMMENDATIONS command from the command line processor (CLP). The command syntax supports querying recommendations to resolve a specific health alert, such as a health indicator that entered an alert state on a particular object.

### Before you begin

You must have an instance attachment to retrieve recommendations from the health monitor. If there is not an attachment to an instance, a default instance attachment is created. To obtain recommendations from a health monitor on a remote instance, you must first attach to that instance. No special authority is required to retrieve recommendations from the health monitor.

### About this task

The command syntax also supports retrieval of the complete set of recommendations for a specific health indicator, which does not have to be in an alert state when the command is executed. Recommendations for resolving an alert on a specific health indicator can be queried at either a single partition level or a global level.

When querying recommendations on a health alert on a specific object, the health monitor is solving a specific alert and is able to provide details on the alert that is being resolved in the problem section of the output.

The health monitor is also able to provide a ranking for the recommendations and, in some cases, it might be able to generate scripts that can be executed to resolve the alert. Additionally, the health monitor might reject and not display some recommendations if they are not applicable to the particular problem situation. Alternatively, if recommendations are queried by health indicator name only, as in the first example shown, the total set of possible recommendations is always returned. In such cases, the CLP command is simply providing information about actions that you should consider undertaking if they see an alert.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

## Procedure

Retrieve the recommendations using the GET RECOMMENDATIONS command:

- You might want to issue the following command to see the total set of actions that could be recommended to resolve an alert on the **db.db\_op\_status** health indicator.

```
db2 get recommendations for health indicator db.db_op_status
```

In this example, the full set of recommendations for the **db.db\_op\_status** health indicator is returned. The health indicator does not have to be in an alert state to issue this command.

This output shows that there are two possible recommendations for this health indicator: unquiesce the database or investigate rollforward progress on the database. Because the command is being used to query all possible recommendations, rather than to ask how to resolve a specific alert, the health monitor cannot identify the best recommendation in this case. As a result, the full set of recommendations is returned.

Recommendations:

Recommendation: Investigate rollforward progress.

A rollforward is in progress on the database due to an explicit request from the administrator. You have to wait for the rollforward to complete for the instance to return to active state.

From the Command Line Processor, issue the commands shown in the following example to view the progress of the rollforward utility:

```
LIST UTILITIES SHOW DETAIL
```

Recommendation: Unquiesce the database.

The database has been put into QUIESCE PENDING or QUIESCE state by an explicit request from the administrator. If you have QUIESCE\_CONNECT authority, or are DBADM or SYSADM, you will still have access to the database and will be able to use it normally. For all other users, new connections to the database are not permitted and new units of work cannot be started. Also, depending on the quiesce request, active

units of work will be allowed to complete or will be rolled back immediately. You can issue an unquiesce to return to active state.

From the Command Line Processor, issue the commands shown in the following example:

```
CONNECT TO DATABASE database-alias
UNQUIESCE DATABASE
```

- Suppose you observe that the health indicator **db.db\_heap\_util** entered an alert state for the database **SAMPLE**, and you want to determine how to resolve the alert. In this case, you want to resolve a specific problem, therefore you could issue the **GET RECOMMENDATIONS** command in the following way:

```
db2 get recommendations for health indicator db.db_heap_util
    for database on sample
```

This output shows a summary of the problem and a set of recommendations to resolve the problem. The health monitor ranked the recommendations in its order of preference. Each recommendation contains a description and a set of actions that indicate how to perform the recommended action.

Problem:

|                        |                       |
|------------------------|-----------------------|
| Indicator Name         | = db.db_heap_util     |
| Value                  | = 42                  |
| Evaluation timestamp   | = 11/25/2003 19:04:54 |
| Alert state            | = Alarm               |
| Additional information | =                     |

Recommendations:

Recommendation: Increase the database heap size.

Rank: 1

Increase the database configuration parameter **dbheap** sufficiently to move utilization to normal operating levels. To increase the value, set the new value of **dbheap** to be equal to  $(\text{pool\_cur\_size} / (4096 * U))$  where **U** is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then  $U = 0.6 * 0.75 = 0.45$  (or 45%).

Execute the following commands at the DB2 server (this can be done using the **EXEC\_DB2\_CMD** stored procedure):

```
CONNECT TO DATABASE SAMPLE;
UPDATE DB CFG USING DBHEAP 149333;
CONNECT RESET;
```

Recommendation: Investigate memory usage of database heap.

Rank: 2

There is one database heap per database and the database manager uses it on behalf of all applications connected to the database. The data area is expanded as needed up to the maximum specified by **dbheap**.

For more information on the database heap, refer to the DB2 Information Center.

Investigate the amount of memory that was used for the database heap over time to determine the most appropriate value for the database heap configuration parameter. The database system monitor tracks the highest amount of memory that was used for the database heap.

- For partitioned database systems, you can query recommendations for a health indicator that entered an alert state on a certain partition, or globally for all partitions. When recommendations are queried globally, a set of recommendations is returned that applies to the health indicator on all

partitions. For example, if the health indicator is in an alert state on partitions 1 and 3, a collection of two scripts might be returned where each script is to be applied to a different partition.

The following example shows how to query recommendations for a health indicator on a specific partition (in this example, partition number 2):

```
db2 get recommendations for health indicator db.db_heap_util
for database on sample at dbpartitionnum 2
```

The following example shows how to retrieve a set of recommendations to resolve a health indicator that is in an alert state on several partitions:

```
db2 get recommendations for health indicator db.db_heap_util
for database on sample global
```

### Retrieving health recommendations using a client application:

Recommendations can be queried using the `db2GetRecommendations` API in a C or C++ application.

#### Before you begin

You must have an instance attachment to capture a health snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To query recommendations on a remote instance, you must first attach to that instance.

#### About this task

When using the `db2GetRecommendations` API, recommendations are returned in an XML document that is:

- Formatted according to the health recommendations XML schema `DB2RecommendationSchema.xsd` located in the `MISC` subdirectory within the `SQLLIB` directory.
- Encoded in UTF-8 and contains text in the client language.
- Organized as a collection of recommendation sets, where each recommendation set describes a problem (health indicator) being resolved and contains one or more recommendations to resolve that health indicator. Refer to the schema definition for specific details about what information that can be retrieved from the document.

All information available through the CLP is also available in the XML recommendation document that is returned.

To retrieve health recommendations using a client application:

#### Procedure

1. Include the `sqlmon.h` and `db2ApiDf.h` DB2 header files. These are found in the `sqllib\include` directory.

```
#include <db2ApiDf.h>
#include <sqlmon.h>
```

2. Declare the `sqlca`, and the `db2GetRecommendationsData` structure.

```
struct sqlca sqlca ;
db2GetRecommendationsData recData ;

memset( &sqlca, '\0', sizeof( struct sqlca ) ) ;
memset( &recData, '\0', sizeof( db2GetRecommendationsData ) ) ;
```

3. Populate the `db2GetRecommendationsData` structure with information about the alert for which you want to retrieve recommendations. In the code excerpt that follows, recommendations are being queried for the `db2.db_heap_util` health indicator on the Sample database.

```
recData.iSchemaVersion = DB2HEALTH_RECSCHEMA_VERSION8_2 ;
recData.iNodeNumber = SQLM_CURRENT_NODE ;
recData.iIndicatorID = SQLM_HI_DATABASE_HEAP_UTILIZATION ;
recData.iObjType = DB2HEALTH_OBJTYPE_DATABASE ;
recData.piDbName = "SAMPLE" ;
```

4. Invoke the `db2GetRecommendations` API to retrieve recommendations for an alert on this health indicator on the specified database.

```
db2GetRecommendations( db2Version820, &recData, &sqlca ) ;
```

5. Check the `sqlcode` returned in the `sqlca` for any errors that occurred. If the API call was successful, process the recommendation XML document that is returned in the `poRecommendation` field of the `db2GetRecommendationsData` structure. Use your choice of XML parser to extract the required elements or attributes. Refer to the `DB2RecommendationSchema.xsd` XML schema in the `sql11b\misc` directory for details about the information that can be retrieved from the XML document.

6. Free any memory allocated by the `db2GetRecommendations` API. This will free the recommendation document returned in the `poRecommendation` field of the `db2GetRecommendationsData` structure.

```
db2GetRecommendationsFree( db2Version820, &recData, &sqlca ) ;
```

## Results

Typically you would combine the preceding code with a call to the snapshot APIs to take a health snapshot because recommendations are generally queried when you detect a health indicator has entered an alert state.

## Health indicator configuration

A default health monitor configuration is provided during installation. This configuration ensures that the health monitor can evaluate the health of the database environment as soon as DB2 is started. However, the health monitor's behavior in evaluating health indicators and reacting to alert states can be fine-tuned through configuration for a specific user's environment

The health monitor SQL table functions have been deprecated.

**Important:** The health monitor, health indicators, and related components have been deprecated in Version 9.7 and might be removed in a future release. Health monitor is not supported in DB2 pureScale environments. For more information, see the “Health monitor has been deprecated” topic at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

There are different levels at which the configuration can be defined. A default configuration of factory settings is provided for each health indicator when DB2 is installed. When the health monitor starts for the first time, a copy of the factory settings provides the defaults for the instance and global settings.

Instance settings apply to the instance. Global settings apply to objects such as databases, table spaces, and table space containers in the instance that do not have customized settings defined.

Updating health indicator settings for a specific database, table space, or table space container creates object settings for the updated health indicators. The default for object settings is the global settings.

The health monitor checks the object settings when it processes a health indicator for a particular database, table space, or table space container. If the settings for a particular health indicator have never been updated, the default global settings are used to process the health indicator. The instance settings are used when the health monitor processes a health indicator for the instance.

You can alter health monitor behavior by using a number of attributes that can be configured for each health indicator. The first set of parameters (evaluation flag, thresholds, sensitivity) defines when the health monitor will generate an alert for a health indicator. The second set of parameters (action flag, actions) defines what the health monitor does upon generating the alert.

#### **Evaluation flag**

Each health indicator has an evaluation flag to enable or disable evaluation of alert state.

#### **Warning and alarm thresholds**

Threshold-based health indicators have settings defining the warning and alarm regions for the health indicator value. These warning and alarm threshold values can be modified for your particular database environment.

#### **Sensitivity parameter**

The sensitivity parameter defines the minimum amount of time, in seconds, that the health indicator value has to be in an alert state before the alert is generated. The wait time associated with the sensitivity value starts on the first refresh interval during which the health indicator value enters an alert state. You can use this value to eliminate erroneous alerts generated due to temporary spikes in resource usage.

Consider an example using the Log Utilization (*db.log\_util*) health indicator. Suppose that you review the DB2 notify log on a weekly basis. In the first week, an entry for *db.log\_util* is in alarm state. You recall having received notification for this situation, but on checking for the alert situation from the CLP, the health indicator was back in normal state. After a second week, you notice a second alarm notification entry for the same health indicator at the same time of the week. You investigate activity in your database environment on the two occasions that alerts were generated, and you discover that an application that takes a long time to commit is run weekly. This application causes the log utilization to spike for a short time, approximately eight to nine minutes, until the application commits. You can see from the history entries in the alarm notification record in the notification log, that the *db.log\_util* health indicator is evaluated every 10 minutes. Because the alert is being generated, the application time must be spanning that refresh interval. You set the sensitivity for the *db.log\_util* parameter to ten minutes. Now every time the value of *db.log\_util* first enters the warning or alarm threshold regions, the value must remain in that region for at least ten minutes before the alert is generated. No further notification entries are recorded in the notification log for this situation because the application only lasts eight to nine minutes.

#### **Action flag**

The running of actions on alert generation is controlled by the action flag. Only when the action flag is enabled are the configured alert actions run.

## Actions

Script or task actions can be configured to run on alert occurrence. For threshold-based health indicators, actions can be configured to run on warning or alarm thresholds. For state-based health indicators, actions can be configured to run on any of the possible non-normal conditions. The DB2 administration server must be running for actions to run.

The following input parameters are passed to every operating system command script:

- <health indicator shortname>
- <object name>
- <value | state>
- <alert type>

Script actions use the default interpreter on the operating system. If you want to use a non-default interpreter, create a task using the ADMIN\_TASK\_ADD procedure with the script content. In a multipartitioned environment, the script defined in the script action must be accessible by all partitions.

The refresh interval at which the health monitor checks each health indicator cannot be configured. The recommendation actions considered by the health monitor cannot be configured.

The health monitor configuration is stored in a binary file, HealthRules.reg:

- On Windows, HealthRules.reg is stored in x:\<SQLLIB\_PATH>\<INSTANCE\_NAME>. For example, d:\sql11ib\DB2.
- On UNIX, HealthRules.reg is stored in ~/<SQLLIB\_PATH>/cfg. For example, ~/home/sql11ib/cfg.

It is possible to replicate a health monitor configuration to other DB2 Version 8 instances on a Linux, UNIX, or Windows server. You can accomplish this replication by copying the binary configuration file to the appropriate directory location on the target instance.

## Retrieving health indicator configuration using the CLP:

The GET ALERT CONFIGURATION command allows you to view the factory settings and the instance, global, and object settings.

### Procedure

1. To view the global settings for database-level health indicators, which apply to all databases without customized settings for the health indicator, issue the following command:

```
DB2 GET ALERT CONFIGURATION FOR DATABASES
```

2. To view the global settings for database-level health indicators, which apply to all databases without customized settings for the health indicator, issue the following command:

```
DB2 GET ALERT CONFIGURATION FOR DATABASES
```

The output of each health indicator's settings indicates whether it has been changed from its default. In the following output, the global settings have not been updated; therefore, they are the same as the default factory settings. To view factory settings for database-level health indicators, issue the same command as in the preceding example with the DEFAULT keyword.

#### Alert Configuration

```
Indicator Name      = db.db_op_status
Default            = Yes
Type               = State-based
Sensitivity         = 0
Formula            = db.db_status;
Actions            = Disabled
Threshold or State checking = Enabled

Indicator Name      = db.sort_shrmem_util
Default            = Yes
Type               = Threshold-based
Warning            = 70
Alarm              = 85
Unit               = %
Sensitivity         = 0
Formula            = ((db.sort_shrheap_allocated/sheapthres_shr)
                    *100);
Actions            = Disabled
Threshold or State checking = Enabled
...

```

3. To view the custom settings for the SAMPLE database, issue the following command:

```
DB2 GET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
```

If there are no specific settings for a particular health indicator on the object specified, the global settings for all databases are displayed. To view the settings for a particular health indicator, add the `USING health-indicator-name` clause to any of the preceding examples.

**Health indicator configuration updates using the CLP:** The health indicator configuration for a particular health indicator can be updated for the global settings or the object settings for a particular object.

The `UPDATE ALERT CONFIGURATION` command has four sub-clauses that cover the different update options. Only one sub-clause can be used in each `UPDATE ALERT CONFIGURATION` command. To use more than one of the options, multiple `UPDATE ALERT CONFIGURATION` commands must be issued.

The first sub-clause, `SET parameter-name value`, provides support to update:

- The evaluation flag
- The warning and alarm thresholds (if applicable)
- The sensitivity flag
- The action flag

The corresponding parameter names for these settings are:

- `THRESHOLDSCHECKED`
- `WARNING` and `ALARM`
- `SENSITIVITY`
- `ACTIONSENABLED`

The other three sub-clauses provide support to add, to update, and to delete script or task actions.

The following commands update a threshold-based health indicator configuration for the `db.spilled_sorts` health indicator on the SAMPLE database. The update changes the warning threshold to 25, to enable actions, and to add a script action:

```

DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
SET WARNING 25, ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
ADD ACTION SCRIPT c:\myscript TYPE OS COMMAND LINE PARAMETERS 'space'
WORKING DIRECTORY c:\ ON ALARM USER dba1 PASSWORD dba1

```

The following commands update a state-based health indicator configuration for the *ts.ts\_util* health indicator for the global settings. The update defines an action to run when any table space is in backup pending state.

```

DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
SET ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
ADD ACTION TASK 0.1 ON ATTENTION 32 ON localhost USER dba1 PASSWORD dba1

```

This update will apply to all table spaces for the instance that do not have customized settings for this health indicator.

When adding actions to a health indication configuration, the options for the *ON condition* clause are based on the type of health indicator:

- For a threshold-based health indicator, WARNING and ALARM are valid conditions.
- For a state-based health indicator, the ON ATTENTION *state* option must be used. A valid numeric state, as defined for the health indicator, should be used. The database manager and database operational state values can be found in `sqllib\include\sqlmon.h`. The table space and table space container operational values are listed in `sqllib\include\sqlutil.h`. Note that actions cannot be executed for the database manager down state. Refer to the description of the `db2.db2_op_status` health indicator for details.

### Resetting health indicator configuration using the CLP:

The CLP provides support for the global settings to be reset to the factory settings. The object settings for a particular object can also be reset to the custom settings for that object type.

#### Procedure

- To reset the object settings for the SAMPLE database to the current global settings for databases:

```
DB2 RESET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
```
- Issue the following command to reset the global settings for databases to the factory settings:

```
DB2 RESET ALERT CONFIGURATION FOR DATABASES
```
- To reset the configuration for a particular health indicator, add the USING *health-indicator-name* clause to any of the preceding examples.

### Configuring health indicators using a client application:

Health monitor configuration is accessible through the `db2GetAlertCfg`, `db2UpdateAlertCfg`, and `db2ResetAlertCfg` APIs in a C or C++ application. Each of these APIs can access the factory, instance, global, and object settings.

#### Before you begin

You must have an instance attachment to access the health monitor configuration. If there is not an attachment to an instance, then a default instance attachment is created. To access the health monitor configuration of a remote instance, you must first attach to that instance.

## About this task

Combinations of the **objType** and **defaultType** parameters in the db2GetAlertCfgData structure allow access to the various levels of health indicator configuration.

Table 142. Settings for objType and defaultType to access configuration levels

| Setting          | objType and defaultType   |
|------------------|---|
| Factory settings | objType = DB2ALERTCFG_OBJTYPE_{DBM   DATABASES   TABLESPACES   CONTAINERS} and defaultType = DB2ALERTCFG_DEFAULT  |
| Global settings  | objType = DB2ALERTCFG_OBJTYPE_{DBM   DATABASES   TABLESPACES   CONTAINERS} and defaultType = DB2ALERTCFG_NOT_DEFAULT<br><br>or<br><br>objType = DB2ALERTCFG_OBJTYPE_{DATABASE   TABLESPACE   CONTAINER} and defaultType = DB2ALERTCFG_DEFAULT |
| Object settings  | objType = DB2ALERTCFG_OBJTYPE_{DATABASE   TABLESPACE   CONTAINER} and defaultType = DB2ALERTCFG_NOT_DEFAULT   |

## Procedure

1. To get the specific object setting for health indicators on the SAMPLE database:

- a. Include the db2ApiDf.h DB2 header file, found in the sql\lib\include directory.

```
#include <db2ApiDf.h>
```

- b. Declare and initialize the sqlca and db2GetAlertCfgData structures.

```
struct sqlca ca;  
memset (&sqlca, '\0', sizeof(struct sqlca));
```

```
char* objName = NULL;  
char* dbName = "SAMPLE";  
db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASE;  
db2Uint32 defaultType = DB2ALERTCFG_NOT_DEFAULT;
```

```
db2GetAlertCfgData data = {objType, objName, defaultType, dbName, 0, NULL} ;
```

- c. Call the db2GetAlertCfg API.

```
rc = db2GetAlertCfg (db2Version810, &data, &ca);
```

- d. Process the returned configuration and free the buffer allotted by the API.

```
if (rc >= SQL0_OK) {  
    if ((data.ioNumIndicators > 0) && (data.pioIndicators != NULL)) {  
        db2GetAlertCfgInd *pIndicators = data.pioIndicators;  
  
        for (db2Uint32 i=0; i < data.ioNumIndicators; i++) {  
            //process the entry as necessary using fields defined in db2ApiDf.h  
        }  
    }  
  
    db2GetAlertCfgFree (db2Version810, &data, &ca);  
}
```

2. The following steps detail the procedure to update the alert configuration of the **db.sort\_shrmem\_util** health indicator for the global settings for database objects, setting warning threshold to 80 and adding task action 1.1:

- a. Include the db2ApiDf.h DB2 header file, found in the sql\lib\include directory.

```
#include <db2ApiDf.h>
```

- b. Declare and initialize the sqlca and db2AlertTaskAction structures.

```
struct sqlca ca;
memset (&sqlca, '\0', sizeof(struct sqlca));

db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASES;

db2Uint32 taskCondition = DB2ALERTCFG_CONDITION_WARNING;
char* taskname = "1.1";
char* hostname = NULL;
char* userid = "nobody";
char* password = "nothing";

db2AlertTaskAction newTask={taskname,taskCondition,userid,password,hostname};
```

- c. Declare and initialize the db2UpdateAlertCfgData structure.

```
struct db2UpdateAlertCfgData setData;

setData.iObjType = objType;
setData.piObjName = NULL;
setData.piDbName = NULL;

setData.iIndicatorID = 1002;

setData.iNumIndAttribUpdates = 1;
setData.piIndAttribUpdates[0].iAttribID = DB2ALERTCFG_WARNING;
setData.piIndAttribUpdates[0].piAttribValue == 80;

setData.iNumActionUpdates = 0;
setData.piActionUpdates = NULL;

setData.iNumActionDeletes = 0;
setData.piActionDeletes = NULL;

setData.iNumNewActions = 1;
setData.piNewActions[0].iActionType = DB2ALERTCFG_ACTIONTYPE_TASK;
setData.piNewActions[0].piScriptAttribs = NULL;
setData.piNewActions[0].piTaskAttribs = &newTask;
```

- d. Call the db2UpdateAlertCfg API.

```
rc = db2UpdateAlertCfg(db2Version810, &setData, &ca);
```

3. The following steps detail the procedure to RESET the custom settings for the MYTS table space in the SAMPLE database.

- a. Include the db2ApiDf.h DB2 header file, found in the sqllib\include directory.

```
#include <db2ApiDf.h>
```

- b. Declare and initialize the sqlca and db2ResetAlertCfgData structures.

```
struct sqlca ca;
memset (&sqlca, '\0', sizeof(struct sqlca));

char* objName = "MYTS";
char* dbName = "SAMPLE";
db2Uint32 objType = DB2ALERTCFG_OBJTYPE_TABLESPACE;

db2ResetAlertCfgData data = {objType, objName, dbName};
```

- c. Call the db2ResetAlertCfg API.

```
rc = db2ResetAlertCfg (db2Version810, &data, &ca);
```

### Health monitor alert actions on combined states:

Alert actions are tasks or scripts that are run when a health indicator goes into an alert state.

Starting in DB2 V9.1, the health monitor alert actions defined for the health indicator **ts.ts\_op\_status** on a single alert state are executed whenever this state

is set for the table space, irrespective of the other combined states. This makes it possible to run alert actions on a specific table space state even when it is set in conjunction with other states.

In the following example, an alert action script1 defined on attention state QUIESCED:share will run, even if the table space state is QUIESCED:share and QUIESCE:update at the same time.

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
set actionsenabled yes')
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention QUIESCED_SHARE on aix1 user guest001 using passwd')

```

In the following example, an alert action defined using a combination of states ( QUIESCED:share + QUIESCED:update = 3 ) is executed if and only if the table space state is both QUIESCED:share and QUIESCED:update.

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
set actionsenabled yes')
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention 3 on aix1 user guest001 using passwd')

```

Starting in DB2 V9.1, health monitor alert actions defined on an object with the same action attributes (name, working directory, command line parameters, host, user and password) run only once, even if it was defined on multiple alert states.

In the following example, the same action is defined on two different alert states. The action is only executed once for a given table space, even if the table space state is in both QUIESCED:share and QUIESCED:update.

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention QUIESCED_SHARE on aix1 user guest001 using passwd')
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention QUIESCED_UPDATE on aix1 user guest001 using passwd')

```

---

## Introduction to Windows Management Instrumentation (WMI)

There is an industry initiative that establishes management infrastructure standards and provides a way to combine information from various hardware and software management systems. This initiative is called Web-Based Enterprise Management (WBEM). WBEM is based on the Common Information Model (CIM) schema, which is an industry standard driven by the Desktop Management Task Force (DMTF).

Microsoft Windows Management Instrumentation (WMI) is an implementation of the WBEM initiative for supported Windows platforms. WMI is useful in a Windows enterprise network where it reduces the maintenance and cost of managing enterprise network components. WMI provides:

- A consistent model of Windows operation, configuration, and status.
- A COM API to allow access to management information.
- The ability to operate with other Windows management services.

- A flexible and extensible architecture allowing vendors a means of writing other WMI providers to support new devices, applications, and other enhancements.
- The WMI Query Language (WQL) to create detailed queries of the information.
- An API for management application developers to write Visual Basic or Windows Scripting Host (WSH) scripts.

The WMI architecture has two parts:

1. A management infrastructure that includes the CIM Object Manager (CIMOM) and a central storage area for management data called the CIMOM object repository. CIMOM allows applications to have a uniform way to access management data.
2. WMI providers. WMI providers are the intermediaries between CIMOM and managed objects. Using WMI APIs, WMI providers supply CIMOM with data from managed objects, handle requests on behalf of management applications, and generate event notifications.

Windows Management Instrumentation (WMI) providers are standard COM or DCOM servers that function as mediators between managed objects and the CIM Object Manager (CIMOM). If the CIMOM receives a request from a management application for data that is not available from the CIMOM object repository, or for events, the CIMOM forwards the request to the WMI providers. WMI providers supply data, and event notifications, for managed objects that are specific to their particular domain.

## DB2 database system integration with Windows Management Instrumentation

The snapshot monitors can be accessed by Windows Management Instrumentation (WMI) by means of the DB2 performance counters and using the built-in PerfMon provider.

The DB2 profile registry variables can be accessed by WMI by using the built-in Registry provider.

The WMI Software Development Kit (WMI SDK) includes several built-in providers:

- PerfMon provider
- Registry event provider
- Registry provider
- Windows event log provider
- Win32 provider
- WDM provider

The DB2 errors that are in the Event Logs can be accessed by WMI by using the built-in Windows Event Log provider.

DB2 database system has a DB2 WMI Administration provider, and sample WMI script files, to access the following managed objects:

1. Instances of the database server including those instances that are distributed. The following operations can be done:
  - Enumerate instances
  - Configure database manager parameters
  - Start/stop/query the status of the DB2 server service

- Setup or establish communication
2. Databases. The following operations can be done:
    - Enumerate databases
    - Configure database parameters
    - Create/drop databases
    - Backup/restore/roll forward databases

You will need to register the DB2 WMI provider with the system before running WMI applications. Registration is done by entering the following commands:

- `mofcomp %DB2PATH%\bin\db2wmi.mof`  
This command loads the definition of the DB2 WMI schema into the system.
- `regsvr %DB2PATH%\bin\db2wmi.dll`  
This command registers the DB2 WMI provider COM DLL with Windows.

In both commands, `%DB2PATH%` is the path where DB2 is installed. Also, `db2wmi.mof` is the `.MOF` file that contains the DB2 WMI schema definition.

There are several benefits to integrating with the WMI infrastructure:

1. You are able to easily write scripts to manage DB2 servers in a Windows-based environment using the WMI provided tool. Sample Visual Basic (VBS) scripts are provided to carry out simple tasks such as listing instances, creating and dropping databases, and updating configuration parameters. The sample scripts are included in the DB2 Application Development for Windows product.
2. You can create powerful management applications that perform many tasks using WMI. The tasks could include:
  - Displaying system information
  - Monitoring DB2 performance
  - Monitoring DB2 system resource consumption

By monitoring both system events and DB2 events through this type of management application, you can manage the database better.

3. You can use existing COM and Visual Basic programming knowledge and skills. By providing a COM or Visual Basic interface, your programmers can save time when developing enterprise management applications.

## Performance monitoring on Windows platforms

When working with DB2 database manager for Windows, there are tools that can be used to monitor performance.

**Important:** The Windows Performance Monitor has been deprecated and may be discontinued in a future release. In Version 10.1 the IBM InfoSphere Optim Performance Manager should be used instead.

- **Windows Performance Monitor**

The Windows Performance Monitor enables you to monitor both database and system performance, retrieving information from any of the performance data providers registered with the system. Windows also provides performance information data on all aspects of computer operation including:

- CPU usage
- Memory utilization
- Disk activity
- Network activity

## Registering DB2 with the Windows performance monitor

### About this task

The setup program automatically registers DB2 with the Windows Performance Monitor for you.

To make DB2 database and DB2 Connect™ performance information accessible to the Windows Performance Monitor, you must register the DLL for the DB2 for Windows Performance Counters. This also enables any other Windows application using the Win32 performance APIs to get performance data. To install and register the DB2 Performance Counters DLL (DB2Perf.DLL) with the Windows Performance Monitor, type:

```
db2perfi -i
```

Registering the DLL also creates a new key in the services option of the registry. One entry gives the name of the DLL, which provides the counter support. Three other entries give names of functions provided within that DLL. These functions include:

**Open** Called when the DLL is first loaded by the system in a process.

**Collect**

Called to request performance information from the DLL.

**Close** Called when the DLL is unloaded.

### Enabling remote access to DB2 performance information

To see Windows performance objects from another DB2 for Windows computer, you must register an administrator username and password with the DB2 database manager. The default Windows Performance Monitor username, SYSTEM, is a DB2 database reserved word and cannot be used.

### About this task

If your DB2 for Windows workstation is networked to other Windows computers, you can use the feature described in this section.

To register the name, type:

```
db2perfr -r username password
```

**Note:** The username used must conform to the DB2 database naming rules.

The username and password data is held in a key in the registry, with security that allows access only by administrators and the SYSTEM account. The data is encoded to prevent security concerns about storing an administrator password in the registry.

**Note:**

1. Once a username and password combination has been registered with the DB2 database system, even local instances of the Performance Monitor will explicitly log on using that username and password. This means that if the username information registered with DB2 database system does not match, local sessions of the Performance Monitor will not show DB2 database performance information.
2. The username and password combination must be maintained to match the username and password values stored in the Windows Security database. If the

username or password is changed in the Windows Security database, the username and password combination used for remote performance monitoring must be reset.

3. To deregister, type:

```
db2perfr -u <username> <password>
```

## Displaying DB2 database and DB2 Connect performance values About this task

To display DB2 database and DB2 Connect performance values using the Performance Monitor, simply choose the performance counters whose values you want displayed from the **Add to** box. This box displays a list of performance objects providing performance data. Select an object to see a list of the counters it supplies.

A performance object can also have multiple instances. For example, the LogicalDisk object provides counters such as “% Disk Read Time” and “Disk Bytes/sec”; it also has an instance for each logical drive in the computer, including “C:” and “D:”.

## Windows performance objects

Windows provides the following performance objects:

- **DB2 Database Manager**

This object provides general information for a single Windows instance. The DB2 database instance being monitored appears as the object instance.

For practical and performance reasons, you can only get performance information from one DB2 database instance at a time. The DB2 database instance that the Performance Monitor shows is governed by the db2instance registry variable in the Performance Monitor process. If you have multiple DB2 database instances running simultaneously and want to see performance information from more than one, you must start a separate session of the Performance Monitor, with db2instance set to the relevant value for each DB2 database instance to be monitored.

If you are running a partitioned database environment, you can only get performance information from one database partition server at a time. By default, the performance information for the default database partition (that is, the database partition that has logical port 0) is displayed. To see performance information of another database partition, you must start a separate session of the Performance Monitor with the DB2NODE environment variable set to the database partition number of the database partition to be monitored.

- **DB2 Databases**

This object provides information for a particular database. Information is available for each currently active database.

- **DB2 Applications**

This object provides information for a particular DB2 database application. Information is available for each currently active DB2 database application.

- **DB2 DCS Databases**

This object provides information for a particular DCS database. Information is available for each currently active database.

- **DB2 DCS Applications**

This object provides information for a particular DB2 DCS application. Information is available for each currently active DB2 DCS application.

Which of these objects will be listed by the Windows Performance Monitor depends on what is installed on your Windows computer and what applications are active. For example, if the DB2 database manager is installed has been started, the DB2 Database Manager object will be listed. If there are also some DB2 databases and applications currently active on that computer, the DB2 Databases and DB2 Applications objects will be listed as well. If you are using your Windows system as a DB2 Connect gateway and there are some DCS databases and applications currently active, the DB2 DCS Databases and DB2 DCS Applications objects will be listed.

## Accessing remote DB2 database performance information

### About this task

Enabling remote access to DB2 Performance Information was discussed earlier. In the **Add to** box, select another computer to monitor. This brings up a list of all the available performance objects on that computer.

In order to be able to monitor DB2 Performance object on a remote computer, the level of the DB2 database or DB2 Connect code installed on that computer must be Version 6 or higher.

## Resetting DB2 performance values

### About this task

When an application calls the DB2 monitor APIs, the information returned is normally the cumulative values since the DB2 database server was started. However, often it is useful to:

- Reset performance values
- Run a test
- Reset the values again
- Re-run the test.

To reset database performance values, use the **db2perf** program. Type:

```
db2perf
```

By default, this resets performance values for all active DB2 databases. However, you can also specify a list of databases to reset. You can also use the **-d** option to specify that performance values for DCS databases should be reset. For example:

```
db2perf
db2perf dbalias1 dbalias2 ... dbaliasn

db2perf -d
db2perf -d dbalias1 dbalias2 ... dbaliasn
```

The first example resets performance values for all active DB2 databases. The next example resets values for specific DB2 databases. The third example resets performance values for all active DB2 DCS databases. The last example resets values for specific DB2 DCS databases.

The **db2perf** program resets the values for ALL programs currently accessing database performance information for the relevant DB2 database server instance (that is, the one held in DB2INSTANCE in the session in which you run **db2perf**).

Invoking **db2perf** also resets the values seen by anyone remotely accessing DB2 database performance information when the **db2perf** command is executed.

**Note:** There is a DB2 database API, `sqlmrset`, that allows an application to reset the values it sees locally, not globally, for particular databases.

---

## Part 2. Monitor elements

Monitor elements are data structures used to store information about a particular aspect of the database system status. For example, the monitor element **direct\_reads** reflects the number of read operations that take place that are performed directly from disk, rather than from any buffer pool.

Each monitor element reflects one of the following types of data:

### Counter

Counters track the number of times something happens. For example, the **deadlocks** monitor element records the total number of deadlocks that have occurred. Other examples of counters include **commit\_sql\_stmts** (**commit statements attempted**), **rows\_deleted** and **total\_sorts**.

**Gauge** Gauges reflect a measurement of how much of something is happening or is used. For example, time-spent monitor elements, such as **total\_section\_proc\_time** or **total\_sort\_time** are measures of how much time is used in different phases of processing. Other examples of gauges include: **locks\_held**, **num\_extent\_moved**, and **sort\_heap\_allocated**. Compared to counters, which can only increase over time, the values in gauges might go up or down, depending on what is happening in the database.

### Watermark

Watermarks reflect the highest value reached for a given measurement. For example, **uow\_total\_time\_top** shows the lifetime of the longest-running unit of work since the database was activated. Other examples of watermarks include: **pkg\_cache\_size\_top**, and **sort\_heap\_top**.

**Text** Many monitor elements report text values. For example, **stmt\_text** contains the text of an SQL statement. Other examples of text monitor elements include: **table\_name**, **tablespace\_type**, and **db\_storage\_path\_state**.

### Timestamp

Timestamp monitor elements show the time that something happened. For example, **conn\_time** shows the time that a connection was made to a database. Other examples of timestamp monitor elements include: **lock\_wait\_start\_time**, **stmt\_first\_use\_time**, and **uow\_stop\_time**. Compared to gauges that measure elapsed time, timestamps measure the exact point in time that something begins or ends.

You can examine monitor elements using one or more of the various monitoring interfaces provided with the DB2 product, such as table functions or event monitors..



---

## Chapter 6. Request monitor elements

Request monitor elements, also known as *request metrics*, measure the volume of work or effort expended by the database server to process different types of requests, including overall system processing, requests related to a specific type of processing, and requests related to a specific data server environment.

Use request monitor elements to monitor the database system, specifically the volume of work and the effort expended by the data server to process application requests.

A *request* is a directive to a database agent to perform some work that expends database resources. Sources of the request can include:

- A directive issued directly by an external application, such as an OPEN or EXECUTE directive. These are referred to as application requests.
- A directive issued by a coordinator agent to a subagent at the same or a different database member.
- A directive issued by an agent at a different database member.

Some representative monitor elements for measuring overall system processing information:

- The **rqsts\_completed\_total** monitor element measures the number of completed by the system.
- The **total\_rqst\_time** monitor element measures the time spent by requests in the data server, including wait time and processing time
- The **total\_wait\_time** monitor element measures the overall wait time.
- The **total\_cpu\_time** monitor element measures the CPU usage time.

Some representative monitor elements for measuring client-server processing information:

- The **client\_idle\_wait\_time** monitor element measures the time spent waiting for the next request from an open connection.
- The **tcPIP\_recv\_volume** monitor element measures the volume of data received by the data server from clients over TCP/IP.

Some representative monitor elements for measuring common data server processing operations:

- **pool\_data\_l\_reads** is one of the monitor elements providing information about buffer pool resource usage.
- **pool\_read\_time** is one of the monitor elements providing information about I/O processing.
- **lock\_wait\_time** is one of the monitor elements providing information about locks and locking.
- **total\_section\_sorts** is one of the monitor elements providing information about sorts.

Some representative monitor elements for monitoring processing relevant to selected types of data server environments:

- **fcm\_recv\_wait\_time** is one of the monitor elements measuring fast communications manager (FCM) processing.

- `wlm_queue_time_total` is one of the monitor elements measuring workload management control actions.

## Accessing request metrics using table functions

You can use the following table functions to access the request metrics:

- `MON_GET_SERVICE_SUBCLASS` and `MON_GET_SERVICE_SUBCLASS_DETAILS`
- `MON_GET_WORKLOAD` and `MON_GET_WORKLOAD_DETAILS`
- `MON_GET_CONNECTION` and `MON_GET_CONNECTION_DETAILS`
- `MON_GET_UNIT_OF_WORK` and `MON_GET_UNIT_OF_WORK_DETAILS`

Each table function in this set of monitor table functions has two forms, one of which has a name ending with "DETAILS." The function that does not end with "DETAILS" provides an SQL relational interface that returns the most commonly needed data. The other function provides XML-based access to the monitor data and returns a more comprehensive set of data.

This set of table functions enables you to focus on request metrics at a particular level of aggregation. You can choose the table function that enables you to focus on subset (or aggregation) of the system workload you are interested in a given situation. All of these table functions include a common set of request metric monitor elements. Each table function may return a few additional details not common with all the table functions.

In a database with no user-defined workloads or service classes, all of the user work performed by the database manager occurs in the default user workload and user service class. The table functions that return data for each service class (or workload) return data for a single service class (or workload) that represents the processing for the user workload for the entire database.

In a database with user-defined workloads and service classes, table functions that return data for each service class (or workload) enable you to compare processing per service class (or workload). Using SQL, you can sum the values across service classes (or workloads) to obtain the value of a monitor element that represents the processing for the user workload for the entire database.

## Accessing request metrics using event monitors

Request metrics are reported by the following event monitors:

- Statistics event monitor - Request metrics are one of several types of information reported by this event monitor.
- UoW event monitor - This event monitor reports similar or identical fields as the `MON_GET_UNIT_OF_WORK` table function

---

## Activity monitor elements

Activity monitor elements, also known as *activity metrics*, are a subset of request monitor elements. Use activity metrics to monitor the subset of data server processing related to executing activities, especially processing done to execute SQL statement sections.

Request monitor elements monitor the complete volume of work and effort expended by the data server to process application requests. Activity monitor elements monitor the work done to execute SQL statement sections, including locking, sorting, and row processing.

To access the current values for activity monitor elements, use the following table functions:

#### **MON\_GET\_ACTIVITY\_DETAILS**

Returns details about one or more activities in progress. Specify the activities of interest in the input parameters. Data returned includes activity metric monitor elements, many other monitor elements, and statement text. Data is returned in XML format.

#### **MON\_GET\_PKG\_CACHE\_STMT**

Returns details for some or all SQL statement sections in the database package cache, which includes both static and dynamic SQL statements. Data returned includes activity metric monitor elements aggregated over all executions of the section since it was added to the package cache. Data is returned in a relational form.

Use the activity event monitor to access historical data about activities. This monitor captures data for each execution of each activity. The activity event monitor captures the same activity monitor elements as the `MON_GET_ACTIVITY_DETAILS` table function. It also captures some additional information.



---

## Chapter 7. Data object monitor elements

Data object monitor elements provide information about operations performed on particular data objects, including tables, indexes, buffer pools, table spaces, and containers.

Every data object type has a set of monitor elements that can be monitored. For example, buffer pools have elements that can be used to calculate buffer pool hit ratios.

Use the following table functions to access current values for data object monitor elements. These monitor table functions return data in a relational form:

- MON\_GET\_BUFFERPOOL
- MON\_GET\_TABLESPACE
- MON\_GET\_CONTAINER
- MON\_GET\_TABLE
- MON\_GET\_INDEX



---

## Chapter 8. Monitor element collection levels

The *monitor element collection level* for a monitor element refers to what, if any settings must be active for data to be collected for that element. For many monitor elements, data collection is controlled by configuration parameters, clauses in the DDL used to define workload management objects, or a combination of both.

### Collection level settings

Setting collection levels using database configuration parameters sets the default collection level for a specific class of monitor elements for the entire database. For example, setting the configuration parameter `mon_req_metrics` to `BASE` causes request metrics to be collected for all agents running in the database. For request and activity metrics, and for section actuals monitor elements, you can also specify a collection level for specific WLM objects that is different from the level used for the database as a whole. In this way, the *effective collection level* for a given monitor element is determined by applying the broadest (highest) collection level specified for that element for the scope in which it is collected. See the section titled “Examples” on page 526 for examples of how the different collection scopes work together.

Collection levels are used in the topics that describe most monitor elements. (Monitor elements returned by snapshot interfaces use monitor switches rather than collection levels.) For example, Table 143 shows the table that describes the interfaces that return the monitor element `skipped_prefetch_data_p_reads`, along with the monitor element collection level that must be active for data to be collected for this element.

Most topics include such a table to show what interfaces return the monitor element, and what the minimum collection level must be for data to be collect for the monitor element described in the topic.

Table 143. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

The collection levels used in monitor element reference topics are as follows:

#### Always collected

Data for this monitor element is always collected. There are no configuration parameters, or SQL statement options to control collection of this information.

#### DATA OBJECT METRICS BASE, DATA OBJECT METRICS EXTENDED

Monitor elements with this collection level are collected if the database configuration parameter `mon_obj_metrics` is set to `BASE` or `EXTENDED`. If `mon_obj_metrics` is set to `NONE`, no data is collected.

#### REQUEST METRICS BASE, REQUEST METRICS EXTENDED

Monitor elements with this collection level are collected if the effective

collection level set to BASE or EXTENDED. The effective collection level for request metrics is determined by examining the current setting for the database configuration parameter `mon_req_metrics` and the settings specified for the COLLECT REQUEST METRICS clause for WLM service superclasses.

#### **ACTIVITY METRICS BASE, ACTIVITY METRICS EXTENDED**

Monitor elements with this collection level are collected if the effective collection level set to BASE or EXTENDED. The effective collection level for activity metrics is determined by examining the current setting for the database configuration parameter `mon_act_metrics` and the settings specified for the COLLECT ACTIVITY METRICS clause for WLM workloads.

#### **SECTION ACTUALS BASE**

Monitor elements with this collection level are collected if the effective collection level set to BASE. The effective collection level for section actuals is determined by examining the current setting for the database configuration parameter `section_actuals` along with the settings for the following items:

- The INCLUDE ACTUALS clause as part of the CREATE or ALTER WORKLOAD, CREATE or ALTER WORK ACTION SET, or CREATE or ALTER SERVICE CLASS statements.
- The <collectsectionactuals> setting on the WLM\_SET\_CONN\_ENV routine.

**COLLECT AGGREGATE ACTIVITY DATA, COLLECT AGGREGATE REQUEST DATA** Monitor elements with this collection level are collected if the clause COLLECT AGGREGATE ACTIVITY DATA or COLLECT AGGREGATE REQUEST DATA is included as part of the CREATE or ALTER statement for specific types of WLM objects.

Some elements are marked as “Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function.” In these cases, the element is returned by a function that merely formats the output of event data already collected.

## **Examples**

### **Example 1: Default collection level of NONE for the database as a whole, EXTENDED for a specific service super class.**

This example shows how you can specify a collection level of NONE for, in this case, request metrics monitor elements for the database as a whole, but still collect EXTENDED metrics for agents running in a service super class.

To disable the collection of request metrics for agents running in the database in general, but collect extended metrics for a specific service super class, then perform the following steps:

1. Set the collection level for request metrics for the database as whole to NONE by issuing the following command:  
`DB2 UPDATE DB CFG FOR database-name USING MON_REQ_METRICS NONE`
2. Alter the service super class for which you want to collect request metrics by executing the following statement:  
`ALTER SERVICE CLASS service-class-name COLLECT REQUEST METRICS EXTENDED`

Results: Request metrics are collected for all agents that run in the service super class with the name *service-class-name*. Request metrics are not collected for agents that run outside of that service super class.

**Example 2: Default collection level of EXTENDED for the database as a whole.**

This example shows how the broadest collection level is applied for collecting monitor element data, perhaps with unintended consequences.

- Specify that activity metrics are to be captured for all activities running in the database using the following command:

```
DB2 UPDATE DB CFG FOR database-name USING MON_ACT_METRICS EXTENDED
```

- Modify a WLM workload so that activity metrics are not collected:

```
ALTER WORKLOAD workload-name COLLECT ACTIVITY METRICS NONE
```

Results: Activity metrics are collected for all agents running in the database, including those running as part of the workload *workload-name*. In this case, the effective collection level is determined by the broader collection level (EXTENDED) specified for the database as a whole through the **mon\_act\_metrics** configuration parameter.

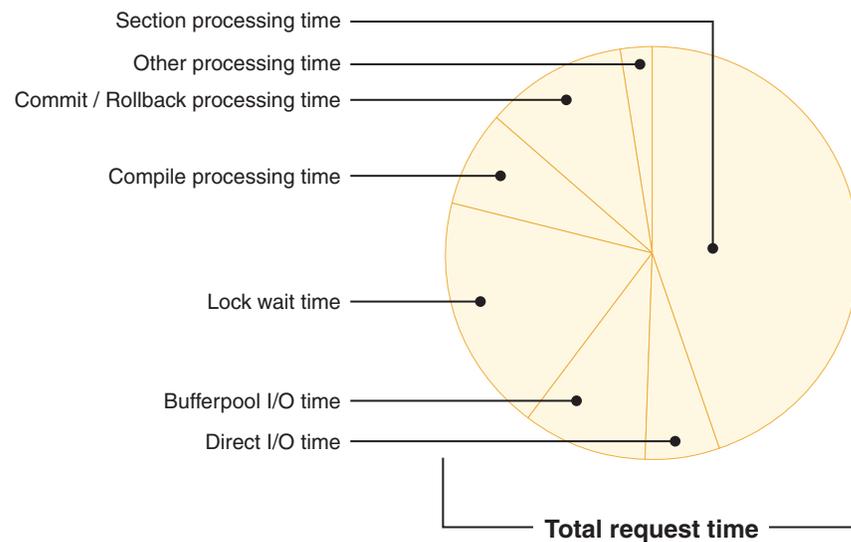


---

## Chapter 9. Time-spent monitor elements

Time-spent monitor elements track how time is spent in the system. You can query them to see where time is spent waiting, or performing different types of processing. You can also view the elapsed time spent in a particular system component.

Figure 11 shows an example of one way that you can view the relative time spent in waits and processing times for a request.



*Figure 11. How time-spent metrics can provide an overall view of where time is spent in the system. Time is broken down into time spent waiting (lock wait time, buffer pool I/O time, direct I/O time, and time spent doing actual processing.*

There are three ways that the database manager monitors time spent in the system:

- Wait times
- Component processing times
- Component elapsed times.

### Wait times

Wait-time monitor elements reflect the time that the database manager spends waiting on something before it can proceed with processing. Some examples include time spent waiting for the following services:

- Incoming client requests
- Locks on objects to be released
- Writing to the diagnostic log
- Reading from or writing to the buffer pool.

Examples of monitor elements that track wait time include `lock_wait_time` and `pool_read_time`.

### Component processing times

These times represent the time spent doing actual processing within a specific logical component of the database. Some examples include time spent doing the following services:

- Committing or rolling back transactions
- Performing database reorganizations
- Compiling SQL
- Loading data
- Performing runstats operations.

Examples of monitor elements that track component processing time include **total\_compile\_proc\_time** and **total\_commit\_proc\_time**.

### Component elapsed times

Component elapsed times reflect the total amount of elapsed time spent within a logical component of the database. They include *both* processing time and various types of wait times that might be incurred during that overall stage of processing. For example, the overall time spent doing a commit includes actual commit processing and might also include various types of wait times, such as time incurred waiting for I/O operations or log file operations to complete.

**Note:** Elapsed time is not the same as elapsed time as measured on a clock; if the overall time spent was split among multiple threads, the time spent in each thread is represented in this number.

Some examples of how you can use component times include:

- Learning where relatively costly processing is taking place for a given workload (for example SQL compilation, as compared to query execution)
- Determining whether the cost of a specific component area can be attributed to actual processing, or whether wait time plays a significant role in reducing throughput
- Understanding the cost of a particular component area (for example, rollback processing) in the context of the overall time spent in the system.

Examples of monitor elements that track overall component times include **total\_compile\_time** and **total\_commit\_time**

You can query component processing times and wait times to get a breakdown of specific wait times relative to processing times. Figure 11 on page 529 is an example of how these two types of time-spent metrics can be viewed relative to one another.

While component elapsed times cannot be used to obtain a breakdown of specific types of wait times (for example, lock waits, I/O-related waits), they do provide an alternative view that you can use to view processing times relative to the overall time spent in a given logical database component. An example would be examining the ratio of actual processing time for table or index reorganizations (**total\_reorg\_proc\_time**), to the overall elapsed time spent performing reorgs (**total\_reorg\_time**), which could include time spent on a variety of miscellaneous processing and waits not directly related to the reorg itself.

---

## Time-spent monitor element hierarchy

The information in many time-spent monitor elements is rolled up in more general monitor elements.

For example, individual wait time elements, such as that for the time spent waiting to receive the information in the next buffer from a table queue (**fc<sub>m</sub>\_tq\_recv\_wait\_time**) and time spent waiting for an FCM reply message (**fc<sub>m</sub>\_message\_recv\_wait\_time**) are both included in overall **fc<sub>m</sub>\_recv\_wait\_time** element. The hierarchical organization of time-spent monitor element makes it possible to choose the element with the most appropriate level of specificity for your needs.

## Dimensions and perspectives for viewing time-spent monitor elements

There are different ways that you can look at the hierarchies of time-spent monitor elements. One way is to look at them from the viewpoint of the system as a whole; you can also look at them in the context of specific activities within the system.

The system-level view or system *dimension* includes elements that you can use to see what the system is doing as a whole. You can also use elements in the system dimension to view time-spent information for specific workloads.

The activity-level view or activity *dimension* includes elements that you can use to see which specific activities the system is spending time on, such as the execution of SQL statements. All monitor elements in the activity dimension are included in the higher-level system dimension.

Within each of these two dimensions are two different *perspectives* that you can use to look at time-spent monitor elements:

- Component processing times as compared to wait times
- Component elapsed times as compared to component processing times

In the first perspective, the values for wait-time elements are independent of and complementary to the values for component processing time elements. If you add the sum of all of the reported wait times to the sum of all component processing times, the resulting value is very close to the value that is reported by the **total\_rqst\_time** monitor element. Any minor difference between the two values is the result of a small amount of miscellaneous component processing time that is not tracked by any monitor element.

In the second perspective, component elapsed times are a superset of component processing times. For example, for a logical component of the database such as that component that performs commits, the total amount of commit processing time, as reported by the **total\_commit\_proc\_time** monitor element, is included in the overall elapsed time for the commit, as reported by the **total\_commit\_time** monitor element. The difference between the total elapsed time and the total processing time is made up of miscellaneous wait or processing times that are not individually tracked by the component elapsed time monitor element.

Viewing component elapsed times relative to wait times is not meaningful, because component elapsed times already include wait times that are incurred as part of the elapsed time in that part of the system. If you created a pie chart consisting of both component elapsed times and wait times, it would not be an accurate representation of the time that is spent in your system because you would be double-counting the various types of wait times.

The sections that follow describe the various dimensions (system and activity) and perspectives (component processing, wait times and component elapsed, and component processing times) from which you can view the information in time-spent monitor elements.

**Tip:** Not all information from time-spent elements is reported through all interfaces. For example, the **client\_idle\_wait\_time** monitor element is applicable only to system-level interfaces such as the `MON_GET_SERVICE_SUBCLASS` table function. See the reference topic for each monitor element for a list of the interfaces that report the element.

- “System dimension”
- “Activity dimension” on page 535

## **System dimension**

Figure 12 on page 533 shows an overall view of how the monitor elements for wait time and component processing time relate to one another, as viewed from the system dimension.

- “client\_idle\_wait\_time - Client idle wait time monitor element” on page 712
- “total\_rqst\_time - Total request time monitor element” on page 1362
  - “total\_wait\_time - Total wait time monitor element” on page 1386
    - “agent\_wait\_time - Agent wait time monitor element” on page 654
    - “wlm\_queue\_time\_total - Workload manager total queue time monitor element” on page 1422
    - “lock\_wait\_time - Time waited on locks monitor element” on page 924
    - “log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933
    - “log\_disk\_wait\_time - Log disk wait time monitor element” on page 935
    - “tcpip\_rcv\_wait\_time - TCP/IP received wait time monitor element” on page 1298
    - “tcpip\_send\_wait\_time - TCP/IP send wait time monitor element” on page 1301
    - “ipc\_rcv\_wait\_time - Interprocess communication received wait time monitor element” on page 888
    - “ipc\_send\_wait\_time - Interprocess communication send wait time monitor element” on page 891
    - “fcm\_rcv\_wait\_time - FCM received wait time monitor element” on page 832<sup>1</sup>
      - “fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element” on page 840<sup>1</sup>
      - “fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element” on page 824<sup>1</sup>
    - “fcm\_send\_wait\_time - FCM send wait time monitor element” on page 836<sup>1</sup>
      - “fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element” on page 844<sup>1</sup>
      - “fcm\_message\_send\_wait\_time - FCM message send wait time monitor element” on page 828<sup>1</sup>
  - “audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683
  - “audit\_file\_write\_wait\_time - Audit file write wait time monitor element” on page 680
  - “diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element” on page 788
  - “pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113
  - “pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128
  - “direct\_read\_time - Direct read time monitor element” on page 792
  - “direct\_write\_time - Direct write time monitor element” on page 798
  - “evmon\_wait\_time - Event monitor wait time monitor element” on page 814
  - “total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335
  - “prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151
  - “comm\_exit\_wait\_time - Communication buffer exit wait time monitor element” on page 719
  - “total\_compile\_proc\_time - Total compile processing time monitor element” on page 1322
    - “total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381
    - “total\_stats\_fabrication\_proc\_time - Total statistics fabrication processing time monitor element” on page 1377
    - *Other monitor elements*<sup>2</sup>
  - “total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element” on page 1340
  - “total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element” on page 1359
  - “total\_section\_proc\_time - Total section processing time monitor element” on page 1367
    - “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368
    - *Other monitor elements*<sup>2</sup>
  - “total\_commit\_proc\_time - Total commits processing time monitor element” on page 1319
  - “total\_rollback\_proc\_time - Total rollback processing time monitor element” on page 1353
  - “total\_runstats\_proc\_time - Total runtime statistics processing time monitor element” on page 1364
  - “total\_reorg\_proc\_time - Total reorganization processing time monitor element” on page 1350
  - “total\_load\_proc\_time - Total load processing time monitor element” on page 1342
  - “total\_connect\_request\_proc\_time - Total connection or switch user request processing time monitor element” on page 1328
    - “total\_connect\_authentication\_proc\_time - Total connection authentication processing time monitor element” on page 1325
    - *Other monitor elements*<sup>2</sup>
  - *Other monitor elements*<sup>3</sup>

<sup>1</sup>These FCM-related wait times do not yield meaningful information when aggregated across members. For more information, see “Wait times for FCM communications for more information”.

<sup>2</sup>These processing time elements include miscellaneous processing and wait times that are not specifically related to this component. For example, the time that the **total\_section\_proc\_time** element reports includes the time that the **total\_section\_proc\_sort\_time** element reports. The **total\_section\_sort\_proc\_time** element reports the time that is spent performing activities such as table and index scans or joins.

Figure 13 on page 535 shows a detailed view of the monitor elements for the time that is spent in various component areas. Each component time is represented by two different monitor elements:

- One that reports the total amount of processing time in a component or stage of processing
- One that reports the overall elapsed time that is spent in the component. This overall time includes the processing time for the component and any other processing or wait times that might be involved.

- “total\_rqst\_time - Total request time monitor element” on page 1362
  - “total\_compile\_time - Total compile time monitor element” on page 1323
    - “total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378
      - “total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381
    - “total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380
      - “total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381
    - *Other monitor elements*<sup>1</sup>
  - “total\_implicit\_compile\_time - Total implicit compile time monitor element” on page 1341
    - “total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element” on page 1340
    - *Other monitor elements*<sup>1</sup>
  - “total\_routine\_user\_code\_time - Total routine user code time monitor element” on page 1360
    - “total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element” on page 1359
  - “total\_section\_time - Total section time monitor element” on page 1373
    - “total\_section\_sort\_time - Total section sort time monitor element” on page 1370
      - “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368
      - *Other monitor elements*<sup>1</sup>
    - *Other monitor elements*<sup>1</sup>
  - “total\_commit\_time - Total commit time monitor element” on page 1320
    - “total\_commit\_proc\_time - Total commits processing time monitor element” on page 1319
    - *Other monitor elements*<sup>1</sup>
  - “total\_rollback\_time - Total rollback time monitor element” on page 1354
    - “total\_rollback\_proc\_time - Total rollback processing time monitor element” on page 1353
    - *Other monitor elements*<sup>1</sup>
  - “total\_runstats - Total runtime statistics monitor element” on page 1363
    - “total\_runstats\_proc\_time - Total runtime statistics processing time monitor element” on page 1364
    - *Other monitor elements*<sup>1</sup>
  - “total\_reorg\_time - Total reorganization time monitor element” on page 1351
    - “total\_reorg\_proc\_time - Total reorganization processing time monitor element” on page 1350
    - *Other monitor elements*<sup>1</sup>
  - “total\_load\_time - Total load time monitor element” on page 1343
    - “total\_load\_proc\_time - Total load processing time monitor element” on page 1342
    - *Other monitor elements*<sup>1</sup>
  - “total\_connect\_request\_time - Total connection or switch user request time monitor element” on page 1330
    - “total\_connect\_authentication\_time - Total connection or switch user authentication request time monitor element” on page 1327
      - “total\_connect\_authentication\_proc\_time - Total connection authentication processing time monitor element” on page 1325
    - *Other monitor elements*<sup>1</sup>
  - *Other monitor elements*<sup>2</sup>

<sup>1</sup>These monitor elements include one or more different types of wait times.

<sup>2</sup>These monitor elements include a small number of miscellaneous types of time spent (processing and wait times) that are not currently monitored.

Figure 13. Component processing time spent monitor elements - System dimension. The values for indented monitor elements are included in the element that precedes those elements in the next-highest level of the hierarchy.

## Activity dimension

Figure 14 on page 536 shows the monitor elements that you can view for activities from the perspective of wait times as compared to component processing times.

- “stmt\_exec\_time - Statement execution time monitor element” on page 1245
  - “total\_act\_wait\_time - Total activity wait time monitor element” on page 1312 <sup>1</sup>
    - “lock\_wait\_time - Time waited on locks monitor element” on page 924
    - “log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933
    - “log\_disk\_wait\_time - Log disk wait time monitor element” on page 935
    - “fcm\_rcv\_wait\_time - FCM received wait time monitor element” on page 832<sup>2</sup>
      - “fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element” on page 840<sup>2</sup>
      - “fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element” on page 824<sup>2</sup>
    - “fcm\_send\_wait\_time - FCM send wait time monitor element” on page 836<sup>2</sup>
      - “fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element” on page 844<sup>2</sup>
      - “fcm\_message\_send\_wait\_time - FCM message send wait time monitor element” on page 828<sup>2</sup>
    - “audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683
    - “evmon\_wait\_time - Event monitor wait time monitor element” on page 814
    - “audit\_file\_write\_wait\_time - Audit file write wait time monitor element” on page 680
    - “diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element” on page 788
    - “pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113
    - “pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128
    - “direct\_read\_time - Direct read time monitor element” on page 792
    - “direct\_write\_time - Direct write time monitor element” on page 798
    - “total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335
    - “prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151
  - “total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element” on page 1356
    - “total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element” on page 1359
    - *Other monitor elements*<sup>3</sup>
  - “total\_section\_proc\_time - Total section processing time monitor element” on page 1367
    - “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368
    - *Other monitor elements*<sup>3</sup>
  - *Other monitor elements*<sup>4</sup>

<sup>1</sup>This monitor element does not include any wait time that is incurred by nested (child) activities that the statement executes.

<sup>2</sup>These FCM-related wait times do not yield meaningful information when aggregated across members. For more information, see “Wait times for FCM communications”.

<sup>3</sup>Includes miscellaneous processing times that are not specifically related to this component.

<sup>4</sup>Includes a small number of miscellaneous types of time spent (processing and wait times) that are not currently monitored. In addition, this time includes any processing and wait times that child activities incur.

*Figure 14. Wait time spent and component processing time spent monitor elements - Activity dimension.* The values for indented monitor elements are included in the element that precedes those elements in the next-highest level of the hierarchy.

Figure 15 on page 537 shows the monitor elements that you can view for activities from the perspective of component elapsed times, which include component processing times.

- “stmt\_exec\_time - Statement execution time monitor element” on page 1245
  - “total\_section\_time - Total section time monitor element” on page 1373
    - “total\_section\_sort\_time - Total section sort time monitor element” on page 1370
      - “total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368
      - *Other*<sup>1</sup>
    - *Other*<sup>2</sup>
  - “total\_routine\_time - Total routine time monitor element” on page 1357
    - “total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements” on page 1356
      - “total\_routine\_user\_code\_time - Total routine user code time monitor element” on page 1360
      - *Other*<sup>2</sup>
    - *Other*<sup>2</sup>

<sup>1</sup>These monitor elements include one or more different types of wait times.

<sup>2</sup>These processing time elements include miscellaneous processing and wait times that are not specifically related to this component.

*Figure 15. Component elapsed time and component processing time monitor elements - Activity dimension. The values for indented monitor elements are included in the element that precedes those elements in the next-highest level of the hierarchy.*

---

## Wait times for FCM communications

In a multi-partition database, or in an environment where there is intrapartition parallelism, the Fast Communications Manager (FCM) manages communication between different agents working on the same statement, whether those agents are on the same or on different members. All FCM communications involve the potential for wait times as one agent waits for work to be done by another, or for data to be transferred from one to another.

FCM-related wait times do not necessarily indicate that processing is blocked across members; for a given statement, work might be proceeding in parallel or serially on sub-agents across members. The FCM-related wait times show the time that an agent is blocked on a single member waiting for another; however, work might very well be proceeding on the other member.

For example, Agent A on member 0 might be blocked waiting for Agent B on member 1 to read the data being sent to it. If Agent B is busy and does not receive the data from the table queue immediately, Agent A will only be allowed to send a limited amount of data before being forced to wait for an acknowledgement from Agent B before sending the remaining data. This wait time is counted as **fcm\_tq\_send\_wait\_time** by Agent A.

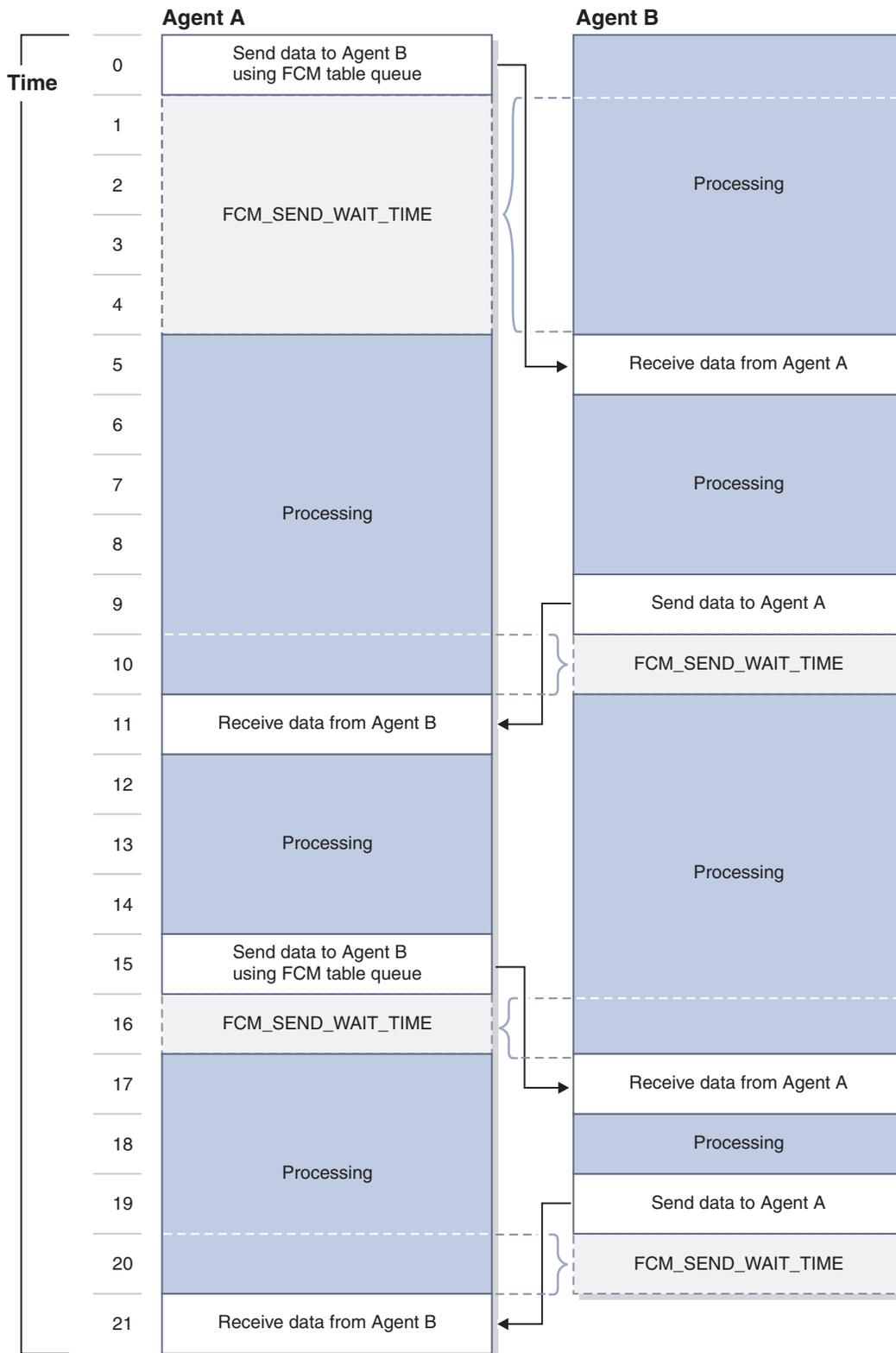


Figure 16. Wait times in FCM communications

Another scenario might involve an agent on one member dispatching a request to an agent on another member. **fcmessage\_recv\_wait\_time** will be incurred if one of the following situations takes place:

- Agent A sends a lengthy request to agent B, and agent B is forced to wait for the full request to be received. In this case, agent B incurs **fcm\_message\_rcv\_wait\_time**.
- Agent A sends a request to agent B, and awaits a reply from agent B. In this case, agent A incurs **fcm\_message\_rcv\_wait\_time**.

**fcm\_message\_send\_wait\_time** will be incurred if one of the following situations takes place:

- Agent A sends a lengthy request to agent B, and is blocked for some reason. For example, Agent A might need to wait while the first portion of the request being sent is processed by the local FCM daemon. In this case, agent A incurs **fcm\_message\_send\_wait\_time**.
- Agent B sends a reply to a request from agent A. If agent B is for some reason blocked before the entire message can be sent, then agent B incurs **fcm\_message\_send\_wait\_time**.

Depending on what it is that you want to measure, it might be appropriate to subtract FCM wait times from the total time spent if you are aggregating metrics for time spent across multiple partitions.

---

## Retrieving and working with time-spent monitor element data

Time-spent monitor element data can be used in almost unlimited ways. For example, you can automate the production of charts that show, at a glance, how time is spent in the system. Or you could use the data to track certain types of wait times in the system over a period of time.

The topics that follow provide some basic examples of how to use the time-spent monitor elements, as well as the table functions that you use to access the data they contain.

### Seeing where time is spent across the system

You can use the time-spent monitor elements to get a view of where time is being spent in the system. Time spent monitor elements can be used to report on specific units of work, service subclasses, workloads or connections.

#### About this task

Once you retrieve the various monitor elements that report where time is being spent in the system, you can view them in several ways. At the most basic level, you can view the reported values as a list. You might want to use the values to create ratios, for example the ratio of lock wait time to total request time. Or you could use the values retrieved to create charts to help you visualize time spent monitor elements relative to another.

#### Notes:

- The values shown in the output for queries are for illustrative purposes only, and should not be construed as representative of what you might see in your own system.
- This task shows you how to retrieve specific time-spent monitor elements. You can also use new formatting functions introduced in Version 9.7 Fix Pack 1 to retrieve time spent monitor elements that meet specific criteria, such as those

with non-zero values, those within a certain range of values that you specify, or the top *n* monitor element (for example, the top five wait times). Example 4 illustrates how these functions work.

## Procedure

1. First, determine what time-spent elements you are interested in. For example, you might want to look at total wait time as compared to the total request time for all connections in your system.
2. Formulate an SQL query that uses one of the monitoring table functions that retrieve the elements you are interested in. In this case, you can retrieve the **total\_request\_time** and the **total\_wait\_time** monitor elements for a connection using the MON\_GET\_CONNECTION table function:

```
SELECT APPLICATION_HANDLE,
       TOTAL_WAIT_TIME,
       TOTAL_RQST_TIME
FROM TABLE(MON_GET_CONNECTION(NULL,NULL))
```

The preceding query will return the following output (times are reported in milliseconds):

| APPLICATION_HANDLE | TOTAL_WAIT_TIME | TOTAL_RQST_TIME |
|--------------------|-----------------|-----------------|
| 39                 | 179             | 269             |
| 78                 | 0               | 0               |
| 51                 | 207             | 316             |
| 77                 | 0               | 21              |
| 50                 | 1014            | 1408            |
| 40                 | 109             | 351             |
| 79                 | 89              | 167             |

7 record(s) selected.

3. In this case, there are 7 application connections; you could use the results from the second and third columns to determine the percentage of time spent waiting for each application. For example, for application 50, compared to the total request time, the wait time spent by that application is  $(1014 \div 1408) \times 100 \approx 72\%$ .

## Example

*Example 1: Determining the average across all connections of the time spent waiting relative to overall request time.*

This example is similar to the preceding one, except that the calculation of the average percentage wait time is done within the SQL:

```
WITH PCTWAIT AS (
  SELECT SUM(TOTAL_WAIT_TIME)AS WAIT_TIME,
         SUM(TOTAL_RQST_TIME)AS RQST_TIME
  FROM TABLE(MON_GET_CONNECTION(NULL,NULL)) AS METRICS)
SELECT WAIT_TIME,
       RQST_TIME,
       CASE WHEN RQST_TIME > 0
            THEN DEC((FLOAT(WAIT_TIME))/FLOAT(RQST_TIME) * 100,5,2)
            ELSE NULL END AS WAIT_PCT FROM PCTWAIT
```

The results of running the preceding query would look something like this:

| WAIT_TIME | RQST_TIME | WAIT_PCT |
|-----------|-----------|----------|
| 1515      | 2439      | 62.11    |

1 record(s) selected.

*Example 2: Comparing total wait time against selected component processing times for a specific service subclass*

This example shows how you can compare time spent in specific types of component processing with the time spent waiting:

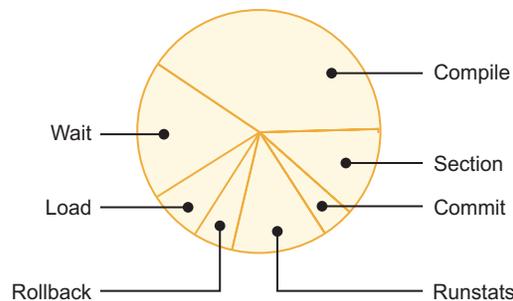
```
SELECT SUM(TOTAL_WAIT_TIME) AS WAIT,
       SUM(TOTAL_COMPILE_PROC_TIME) AS COMPILE,
       SUM(TOTAL_IMPLICIT_COMPILE_PROC_TIME) AS IMP_COMPILE,
       SUM(TOTAL_SECTION_PROC_TIME) AS SECTION,
       SUM(TOTAL_COMMIT_PROC_TIME) AS COMMIT,
       SUM(TOTAL_REORG_PROC_TIME) AS REORG,
       SUM(TOTAL_RUNSTATS_PROC_TIME) AS RUNSTATS,
       SUM(TOTAL_ROLLBACK_PROC_TIME) AS ROLLBACK,
       SUM(TOTAL_LOAD_PROC_TIME) AS LOAD
FROM TABLE(MON_GET_SERVICE_SUBCLASS( 'SYSDEFAULTUSERCLASS', 'SYSDEFAULTSUBCLASS', NULL))
```

The results of the preceding query would look something like this (the output rows from the query have been split for presentation purposes):

| WAIT  | COMPILE  | IMP_COMPILE | SECTION | COMMIT |
|-------|----------|-------------|---------|--------|
| 611   | 1931     | 0           | 395     | 15     |
| REORG | RUNSTATS | ROLLBACK    | LOAD    |        |
| 0     | 432      | 18          | 0       |        |

1 record(s) selected.

The numbers reported could be used to construct a pie chart to show the relative time spent waiting as compared to time spent in different stages of processing (component times of 0 are not included):



*Example 3: View the ratio of total time spent as compared to processing time in different components*

This example shows you how you can get an overview of the time spent doing work in different stages of processing (components), relative to the total time spent in that component. The following query computes the ratio (expressed as a percentage) of the time spent in actual processing as compared to the total elapsed time spent in a specific component.

```

WITH PCTPROC AS (
  SELECT SUM(TOTAL_SECTION_TIME) AS SECT_TIME, SUM(TOTAL_SECTION_PROC_TIME) AS SECT_PROC_TIME,
    SUM(TOTAL_COMPILE_TIME) AS COMP_TIME, SUM(TOTAL_COMPILE_PROC_TIME) AS COMP_PROC_TIME,
    SUM(TOTAL_IMPLICIT_COMPILE_TIME) AS IMP_C_TIME, SUM(TOTAL_IMPLICIT_COMPILE_PROC_TIME) AS IMP_C_PROC_TIME,
    SUM(TOTAL_COMMIT_TIME) AS COMMIT_TIME, SUM(TOTAL_COMMIT_PROC_TIME) AS COMMIT_PROC_TIME,
    SUM(TOTAL_ROLLBACK_TIME) AS ROLLBACK_TIME, SUM(TOTAL_ROLLBACK_PROC_TIME) AS ROLLBACK_PROC_TIME,
    SUM(TOTAL_RUNSTATS_TIME) AS RUNSTATS_TIME, SUM(TOTAL_RUNSTATS_PROC_TIME) AS RUNSTATS_PROC_TIME,
    SUM(TOTAL_REORG_TIME) AS REORG_TIME, SUM(TOTAL_REORG_PROC_TIME) AS REORG_PROC_TIME,
    SUM(TOTAL_LOAD_TIME) AS LOAD_TIME, SUM(TOTAL_LOAD_PROC_TIME) AS LOAD_PROC_TIME
  FROM TABLE(MON_GET_CONNECTION(NULL, -2)) AS METRICS)
  SELECT CASE WHEN SECT_TIME > 0
    THEN DEC((FLOAT(SECT_PROC_TIME) / FLOAT(SECT_TIME)) * 100,5,1)
    ELSE NULL END AS SECT_PROC_PCT,
    CASE WHEN COMP_TIME > 0
    THEN DEC((FLOAT(COMP_PROC_TIME) / FLOAT(COMP_TIME)) * 100,5,1)
    ELSE NULL END AS COMPILE_PROC_PCT,
    CASE WHEN IMP_C_TIME > 0
    THEN DEC((FLOAT(IMP_C_PROC_TIME) / FLOAT(IMP_C_TIME)) * 100,5,1)
    ELSE NULL END AS IMPL_COMPILE_PROC_PCT,
    CASE WHEN ROLLBACK_TIME > 0
    THEN DEC((FLOAT(ROLLBACK_PROC_TIME) / FLOAT(ROLLBACK_TIME)) * 100,5,1)
    ELSE NULL END AS ROLLBACK_PROC_PCT,
    CASE WHEN COMMIT_TIME > 0
    THEN DEC((FLOAT(COMMIT_PROC_TIME) / FLOAT(COMMIT_TIME)) * 100,5,1)
    ELSE NULL END AS COMMIT_PROC_PCT,
    CASE WHEN RUNSTATS_TIME > 0
    THEN DEC((FLOAT(RUNSTATS_PROC_TIME) / FLOAT(RUNSTATS_TIME)) * 100,5,1)
    ELSE NULL END AS RUNSTATS_PROC_PCT,
    CASE WHEN REORG_TIME > 0
    THEN DEC((FLOAT(REORG_PROC_TIME) / FLOAT(REORG_TIME)) * 100,5,1)
    ELSE NULL END AS REORG_PROC_PCT,
    CASE WHEN LOAD_TIME > 0
    THEN DEC((FLOAT(LOAD_PROC_TIME) / FLOAT(LOAD_TIME)) * 100,5,1)
    ELSE NULL END AS LOAD_PROC_PCT
  FROM PCTPROC

```

The query produces the following output:

| SECT_PROC_PCT | COMPILE_PROC_PCT | IMPL_COMPILE_PROC_PCT | ROLLBACK_PROC_PCT | COMMIT_PROC_PCT | RUNSTATS_PROC_PCT | REORG_PROC_PCT | LOAD_PROC_PCT |
|---------------|------------------|-----------------------|-------------------|-----------------|-------------------|----------------|---------------|
| 57.6          | 0.1              | -                     | 96.9              | 95.6            | 0.0               | 71.1           | 84.6          |

1 record(s) selected.

A graphical representation of this data might look something like what is shown in Figure 17:

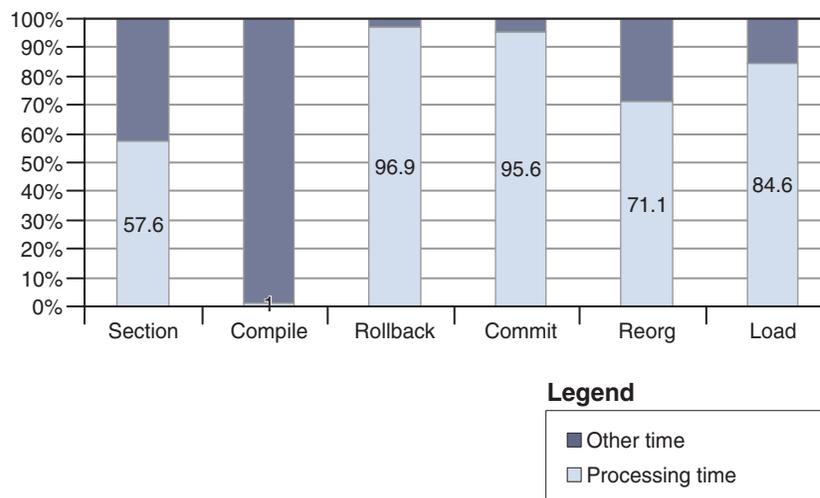


Figure 17. Component processing times as a percentage of overall time spent

**Example 4: Ranking time-spent monitor elements**

In the preceding examples, all monitor elements that are displayed are specified explicitly in the SQL for the query; each appears in its own column in the query results. However, there might be times when you do not know which time spent monitor elements you want to examine, such

as if you want to see the top ten wait-time monitor elements, or only the non-zero time-spent monitor elements.

Several table functions were added in DB2 Version 9.7 Fix Pack 1 that you can use to display monitor elements in a row-oriented format, where each element appears in a row by itself. The table functions you can use to do this have names of the form `MON_FORMAT_XML_*_BY_ROW`. These functions extract metrics from XML documents returned by certain monitoring interfaces. (See “Interfaces that return monitor data in XML documents” on page 13 for more information.)

The `MON_FORMAT_XML_*_BY_ROW` functions are useful when you do not know which elements you want to view. For example, you might want to see the top 10 wait-time monitor elements for the workload named `CLPWORKLOAD`. To collect this information, you can create a statistics event monitor called `DBSTATS` (event\_wlstats logical data group).

Assuming you set up this event monitor to write to a table, it records metrics in a column called `DETAILS_XML`. Once the output table from the event monitor is populated with monitor data, you can construct a query that uses the `MON_FORMAT_XML_WAIT_TIMES_BY_ROW` function to extract the monitor elements you want to see:

```
SELECT SUBSTR(STATS.WORKLOAD_NAME,1,15) AS WORKLOAD_NAME,
       SUBSTR(METRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(METRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE
FROM   WLSTATS_DBSTATS AS STATS,
       TABLE(MON_FORMAT_XML_WAIT_TIMES_BY_ROW(STATS.DETAILS_XML)) AS METRICS
WHERE  WORKLOAD_NAME='CLPWORKLOAD' AND (PARENT_METRIC_NAME='TOTAL_WAIT_TIME')
GROUP BY WORKLOAD_NAME,METRIC_NAME
ORDER BY TOTAL_TIME_VALUE DESC
FETCH FIRST 10 ROWS ONLY
```

**Remember:** Time spent monitor elements are organized into hierarchies. In this example, to avoid double-counting wait times, only the monitor elements that roll-up to `total_wait_time` are included (see the `WHERE` clause in the preceding SQL statement). Otherwise, `total_wait_time` itself would be included in the results, which includes several individual wait times.

The output that follows shows what the results of the preceding query might look like:

| WORKLOAD_NAME | METRIC_NAME             | TOTAL_TIME_VALUE |
|---------------|-------------------------|------------------|
| CLPWORKLOAD   | LOCK_WAIT_TIME          | 15138541         |
| CLPWORKLOAD   | DIRECT_READ_TIME        | 6116231          |
| CLPWORKLOAD   | POOL_READ_TIME          | 6079458          |
| CLPWORKLOAD   | DIRECT_WRITE_TIME       | 452627           |
| CLPWORKLOAD   | POOL_WRITE_TIME         | 386208           |
| CLPWORKLOAD   | IPC_SEND_WAIT_TIME      | 283172           |
| CLPWORKLOAD   | LOG_DISK_WAIT_TIME      | 103888           |
| CLPWORKLOAD   | DIAGLOG_WRITE_WAIT_TIME | 78198            |
| CLPWORKLOAD   | IPC_RECV_WAIT_TIME      | 15612            |
| CLPWORKLOAD   | TCP_IP_SEND_WAIT_TIME   | 3291             |

10 record(s) selected.

## Determining where time is spent during SQL statement execution

One example of retrieving time-spent information at the activity level is viewing time spent monitor elements for specific SQL statements. You can use the `MON_GET_PKG_CACHE_STMT` table function to retrieve this information.

### About this task

This task shows an example of how to retrieve selected time-spent details for SQL statements in the package cache.



STMT\_TEXT

-----  
UPDATE EMPLOYEE SET BONUS=10000 WHERE PERF\_RATING=1  
1 record(s) selected.



---

## Chapter 10. Logical data groups overview

When examining monitor element data, it is frequently useful to look at multiple related elements at the same time. *Logical data groups* are groupings of elements that complement each other.

For example, the uow logical data group includes elements like **appl\_id** (application ID) and **appl\_name** (application name), which are two elements that you can easily imagine wanting to see together.

Logical data groups are used by snapshot monitoring interfaces, and in particular, by event monitors. There are over a thousand monitor elements in the DB2 product. When examining monitor elements, it would be very tiresome to be forced to always have to think about and be forced to specify specifically which elements you want to see. For example, if you are creating an event monitor, thinking about and specifying what elements to capture data for might be very tedious. Instead, the DB2 product associates a default set of logical data groups with each event monitor. This means that you do not need to specify anything in the CREATE EVENT MONITOR statement to capture a useful set of monitor elements; only monitor elements relevant to the events being captured are included. For event monitors that write to regular tables, you have the added flexibility of being able to specify the logical data groups for which you want monitor element data captured.

Event monitors that write to unformatted event (UE) tables capture also capture a default set of monitor elements; when you use the EVMON\_FORMAT\_UE\_TO\_TABLES procedure to generate relational tables, logical data groups are used to group related elements together in separate tables. For example, the lock logical data group contains the elements used in the LOCK\_EVENT table; the participant logical data group contains the elements used in the LOCK\_PARTICIPANT table.

---

### Event monitor logical data groups and monitor elements

Monitor elements that are often useful to examine together are grouped in *logical data groups*.

All event monitors use logical data groups in one way or another. For some event monitor types, you can specify what information you want to collect by specifying the logical data groups for which you want information recorded. Logical data groups are also used to group data together in the output event monitors generate; for example, event monitors that write to tables generally create one table for each logical data group of monitor elements.

The following table lists the logical data groupings and monitor elements that can be returned by event monitoring.

- “changesummary logical data group” on page 42
- “dbdbmcfg logical data group” on page 42
- “ddlstmexec logical data group” on page 42
- “dllock logical data group” on page 43
- “event\_activity logical data group” on page 43
- “event\_activitiymetrics logical data group” on page 45

- “event\_activystmt logical data group” on page 48
- “event\_activityvals logical data group” on page 49
- “event\_bufferpool logical data group” on page 49
- “event\_conn logical data group” on page 51
- “event\_connheader logical data group” on page 53
- “event\_connmemuse logical data group” on page 54
- “event\_data\_value logical data group” on page 54
- “event\_db logical data group” on page 54
- “event\_dbheader logical data group” on page 58
- “event\_dbmemuse logical data group” on page 58
- “event\_deadlock logical data group” on page 59
- “event\_detailed\_dlconn logical data group” on page 59
- “event\_dlconn logical data group” on page 60
- “event\_histogrambin logical data group” on page 61
- “event\_log\_header logical data group” on page 61
- “event\_overflow logical data group” on page 61
- “event\_qstats logical data group” on page 61
- “event\_scstats logical data group” on page 62
- “event\_start logical data group” on page 63
- “event\_stmt logical data group” on page 63
- “event\_stmt\_history logical data group” on page 65
- “event\_subsection logical data group” on page 65
- “event\_table logical data group” on page 66
- “event\_tablespace logical data group” on page 66
- “event\_thresholdviolations logical data group” on page 68
- “event\_wlstats logical data group” on page 69
- “event\_wcstats logical data group” on page 68
- “event\_xact logical data group” on page 70
- “evmonstart logical data group” on page 71
- “lock logical data group” on page 71
- “lock\_participants logical data group” on page 72
- “lock\_participant\_activities logical data group” on page 71
- “lock\_activity\_values logical data group” on page 71
- “pkgcache logical data group” on page 74
- “pkgcache\_metrics logical data group” on page 75
- “pkgcache\_stmt\_args logical data group” on page 79
- “regvar logical data group” on page 79
- “sqlca logical data group” on page 79
- “txncompletion logical data group” on page 79
- “uow logical data group” on page 80
- “uow\_metrics logical data group” on page 81
- “uow\_package\_list logical data group” on page 87
- “uow\_executable\_list logical data group” on page 81
- “utillocation logical data group” on page 87
- “utilphase logical data group” on page 88

- “utilstart logical data group” on page 88
- “utilstop logical data group” on page 88

### **changesummary logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_type - Event Type monitor element” on page 812
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “coord\_member - Coordinator member monitor element” on page 749
- “utility\_invocation\_id - Utility invocation ID” on page 1413
- “utility\_type - Utility Type” on page 1417
- “appl\_id - Application ID monitor element” on page 666
- “appl\_name - Application name monitor element” on page 670
- “application\_handle - Application handle monitor element” on page 675
- “system\_auth\_id - System authorization identifier monitor element” on page 1267
- “session\_auth\_id - Session authorization ID monitor element” on page 1211
- “client\_platform - Client operating platform monitor element” on page 714
- “client\_protocol - Client communication protocol monitor element” on page 716
- “client\_port\_number - Client port number monitor element” on page 715
- “client\_pid - Client process ID monitor element” on page 713
- “client\_hostname - Client hostname monitor element” on page 711
- “client\_wrkstname - Client workstation name monitor element” on page 717
- “client\_acctng - Client accounting string monitor element” on page 709
- “client\_userid - Client user ID monitor element” on page 717
- “client\_applname - Client application name monitor element” on page 710
- “backup\_timestamp - Backup timestamp” on page 689

### **dbdbmcfg logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812
- “cfg\_name - Configuration name” on page 704
- “cfg\_value - Configuration value” on page 706
- “cfg\_value\_flags - Configuration value flags” on page 706
- “cfg\_old\_value - Configuration old value” on page 705
- “cfg\_old\_value\_flags - Configuration old value flags” on page 705
- “cfg\_collection\_type - Configuration collection type” on page 704
- “deferred - Deferred” on page 785

### **ddlstmexec logical data group**

- “event\_id - Event ID monitor element” on page 810
- “event\_timestamp - Event timestamp monitor element” on page 811
- “member - Database member monitor element” on page 957
- “event\_type - Event Type monitor element” on page 812

"global\_transaction\_id - Global transaction identifier monitor element" on page 851  
"local\_transaction\_id - Local transaction identifier monitor element" on page 900  
"savepoint\_id - Savepoint ID" on page 1198  
"uow\_id - Unit of work ID monitor element" on page 1401  
"ddl\_classification - DDL classification" on page 780  
"stmt\_text - SQL statement text monitor element" on page 1255

### **dllock logical data group**

"data\_partition\_id - Data partition identifier monitor element" on page 767  
"lock\_attributes - Lock attributes monitor element" on page 901  
"lock\_count - Lock count monitor element" on page 902  
"lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
"lock\_escalation - Lock escalation monitor element" on page 904  
"lock\_hold\_count - Lock hold count monitor element" on page 911  
"lock\_mode - Lock mode monitor element" on page 912  
"lock\_name - Lock name monitor element" on page 914  
"lock\_object\_name - Lock Object Name" on page 915  
"lock\_object\_type - Lock object type waited on monitor element" on page 916  
"lock\_release\_flags - Lock release flags monitor element" on page 918  
"lock\_status - Lock status monitor element" on page 919  
"node\_number - Node Number" on page 970  
"table\_file\_id - Table file ID monitor element" on page 1268  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"tablespace\_name - Table space name monitor element" on page 1279

**Note:** The underlying implementation for this logical data group is the snapshot monitor LOCK logical data group. If you examine the output for this logical group in the self-describing stream used for the file and pipe output options, you can see that the LOCK group is used to generate the output.

### **event\_activity logical data group**

"act\_exec\_time - Activity execution time monitor element" on page 639  
"activate\_timestamp - Activate timestamp monitor element" on page 644  
"activity\_id - Activity ID monitor element" on page 645  
"activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
"activity\_type - Activity type monitor element" on page 647  
"address - IP address from which the connection was initiated" on page 649  
"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_name - Application name monitor element" on page 670  
"arm\_correlator - Application response measurement correlator monitor element" on page 677  
"coord\_partition\_num - Coordinator partition number monitor element" on page 750

“db\_work\_action\_set\_id - Database work action set ID monitor element” on page 777

“db\_work\_class\_id - Database work class ID monitor element” on page 778

details\_xml (This XML document is a metrics document of type activity\_metrics, as described in the XML schema document sql11ib/misc/DB2MonCommon.xsd.) You can also access the metrics reported in this document through the event\_activitymetrics logical data group.

“intra\_parallel\_state - Current state of intrapartition parallelism monitor element” on page 886

“num\_remaps - Number of remaps monitor element” on page 981

“parent\_activity\_id - Parent activity ID monitor element” on page 1013

“parent\_uow\_id - Parent unit of work ID monitor element” on page 1014

“partial\_record - Partial Record monitor element” on page 1014

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element” on page 1162

“sc\_work\_action\_set\_id - Service class work action set ID monitor element” on page 1198

“sc\_work\_class\_id - Service class work class ID monitor element” on page 1199

“section\_actuals - Section actuals monitor element” on page 1200

“service\_subclass\_name - Service subclass name monitor element” on page 1208

“service\_superclass\_name - Service superclass name monitor element” on page 1209

“session\_auth\_id - Session authorization ID monitor element” on page 1211

“sort\_overflows - Sort overflows monitor element” on page 1224

“sqlca - SQL Communications Area (SQLCA)” on page 1229

“time\_completed - Time completed monitor element” on page 1308

“time\_created - Time created monitor element” on page 1309

“time\_started - Time started monitor element” on page 1310

“total\_sort\_time - Total sort time monitor element” on page 1374

“total\_sorts - Total sorts monitor element” on page 1375

“tpmon\_acc\_str - TP monitor client accounting string monitor element” on page 1387

“tpmon\_client\_app - TP monitor client application name monitor element” on page 1388

“tpmon\_client\_userid - TP monitor client user ID monitor element” on page 1388

“tpmon\_client\_wkstn - TP monitor client workstation name monitor element” on page 1389

“uow\_id - Unit of work ID monitor element” on page 1401

“workload\_id - Workload ID monitor element” on page 1426

“workload\_occurrence\_id - Workload occurrence identifier monitor element” on page 1428

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121

"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123

"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125

"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136

"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139

"prep\_time - Preparation time monitor element" on page 1154

"query\_card\_estimate - Query Number of Rows Estimate" on page 1163

"query\_cost\_estimate - Query cost estimate monitor element" on page 1163

"rows\_fetched - Rows fetched monitor element" on page 1188

"rows\_modified - Rows modified monitor element" on page 1190

"rows\_returned - Rows returned monitor element" on page 1193

"system\_cpu\_time - System CPU time monitor element" on page 1267

"user\_cpu\_time - User CPU time monitor element" on page 1411

"wl\_work\_action\_set\_id - Workload work action set identifier monitor element" on page 1420

"wl\_work\_class\_id - Workload work class identifier monitor element" on page 1421

"mon\_interval\_id - Monitor interval identifier monitor element" on page 967

"member - Database member monitor element" on page 957

"query\_data\_tag\_list - Estimated query data tag list monitor element" on page 1164

"total\_stats\_fabrication\_time - Total statistics fabrication time monitor element" on page 1378

"total\_stats\_fabrications - Total statistics fabrications monitor elements" on page 1379

"total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element" on page 1382

"total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements" on page 1380

### **event\_activitymetrics logical data group**

"audit\_events\_total - Total audit events monitor element" on page 679

"audit\_file\_writes\_total - Total audit files written monitor element" on page 682

"audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element" on page 683

"audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element" on page 685

"coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element" on page 751

"deadlocks - Deadlocks detected monitor element" on page 783

"diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element" on page 788

"diaglog\_writes\_total - Total diagnostic log file writes monitor element" on page 789

"direct\_read\_reqs - Direct read requests monitor element" on page 790

"direct\_read\_time - Direct read time monitor element" on page 792

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_write\_reqs - Direct write requests monitor element" on page 796

"direct\_write\_time - Direct write time monitor element" on page 798

"direct\_writes - Direct writes to database monitor element" on page 800

"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823

"fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element" on page 824

"fcm\_message\_rcvs\_total - Total FCM message receives monitor element" on page 826

"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827

"fcm\_message\_send\_wait\_time - FCM message send wait time monitor element" on page 828

"fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830

"fcm\_rcv\_volume - FCM received volume monitor element" on page 831

"fcm\_rcv\_wait\_time - FCM received wait time monitor element" on page 832

"fcm\_rcvs\_total - FCM receives total monitor element" on page 834

"fcm\_send\_volume - FCM send volume monitor element" on page 835

"fcm\_send\_wait\_time - FCM send wait time monitor element" on page 836

"fcm\_sends\_total - FCM sends total monitor element" on page 838

"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839

"fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element" on page 840

"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842

"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843

"fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element" on page 844

"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846

"lock\_escals - Number of lock escalations monitor element" on page 905

"lock\_timeouts - Number of lock timeouts monitor element" on page 920

"lock\_wait\_time - Time waited on locks monitor element" on page 924

“lock\_waits - Lock waits monitor element” on page 928

“log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933

“log\_disk\_wait\_time - Log disk wait time monitor element” on page 935

“log\_disk\_waits\_total - Total log disk waits monitor element” on page 936

“num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element” on page 975

“num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element” on page 979

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“post\_shrthreshold\_sorts - Post shared threshold sorts monitor element” on page 1144

“post\_threshold\_sorts - Post threshold sorts monitor element” on page 1150

“rows\_modified - Rows modified monitor element” on page 1190

“rows\_read - Rows read monitor element” on page 1191

“rows\_returned - Rows returned monitor element” on page 1193

“sort\_overflows - Sort overflows monitor element” on page 1224

“stmt\_exec\_time - Statement execution time monitor element” on page 1245

"thresh\_violations - Number of threshold violations monitor element" on page 1304  
 "total\_act\_time - Total activity time monitor element" on page 1311  
 "total\_act\_wait\_time - Total activity wait time monitor element" on page 1312  
 "total\_app\_section\_executions - Total application section executions monitor element" on page 1317  
 "total\_cpu\_time - Total CPU time monitor element" on page 1330  
 "total\_routine\_invocations - Total routine invocations monitor elements" on page 1355  
 "total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element" on page 1356  
 "total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements" on page 1356  
 "total\_routine\_time - Total routine time monitor element" on page 1357  
 "total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element" on page 1359  
 "total\_routine\_user\_code\_time - Total routine user code time monitor element" on page 1360  
 "total\_section\_proc\_time - Total section processing time monitor element" on page 1367  
 "total\_section\_sort\_proc\_time - Total section sort processing time monitor element" on page 1368  
 "total\_section\_sort\_time - Total section sort time monitor element" on page 1370  
 "total\_section\_sorts - Total section sorts monitor element" on page 1371  
 "total\_section\_time - Total section time monitor element" on page 1373  
 "total\_sorts - Total sorts monitor element" on page 1375  
 "tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396  
 "wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element" on page 1421  
 "wlm\_queue\_time\_total - Workload manager total queue time monitor element" on page 1422

### **event\_activitystmt logical data group**

"activate\_timestamp - Activate timestamp monitor element" on page 644  
 "activity\_id - Activity ID monitor element" on page 645  
 "activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
 "appl\_id - Application ID monitor element" on page 666  
 "comp\_env\_desc - Compilation environment monitor element" on page 722  
 "creator - Application Creator" on page 763  
 "eff\_stmt\_text - Effective statement text monitor element" on page 805  
 "executable\_id - Executable ID monitor element" on page 818  
 "intra\_parallel\_state - Current state of intrapartition parallelism monitor element" on page 886  
 "package\_name - Package name monitor element" on page 1006  
 "package\_version\_id - Package version monitor element" on page 1007  
 "query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element" on page 1162

"routine\_id - Routine ID monitor element" on page 1187  
"section\_env - Section environment monitor element" on page 1201  
"section\_number - Section number monitor element" on page 1201  
"stmt\_first\_use\_time - Statement first use timestamp monitor element" on page 1246  
"stmt\_invocation\_id - Statement invocation identifier monitor element" on page 1247  
"stmt\_isolation - Statement isolation" on page 1248  
"stmt\_last\_use\_time - Statement last use timestamp monitor element" on page 1248  
"stmt\_lock\_timeout - Statement lock timeout monitor element" on page 1249  
"stmt\_nest\_level - Statement nesting level monitor element" on page 1249  
"stmt\_pkgcache\_id - Statement package cache identifier monitor element" on page 1251  
"stmt\_query\_id - Statement query identifier monitor element" on page 1252  
"stmt\_source\_id - Statement source identifier" on page 1253  
"stmt\_text - SQL statement text monitor element" on page 1255  
"stmt\_type - Statement type monitor element" on page 1256  
"uow\_id - Unit of work ID monitor element" on page 1401  
"member - Database member monitor element" on page 957

### **event\_activityvals logical data group**

"activate\_timestamp - Activate timestamp monitor element" on page 644  
"activity\_id - Activity ID monitor element" on page 645  
"activity\_secondary\_id - Activity secondary ID monitor element" on page 646  
"appl\_id - Application ID monitor element" on page 666  
"intra\_parallel\_state - Current state of intrapartition parallelism monitor element" on page 886  
"query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element" on page 1162  
"stmt\_value\_data - Value data" on page 1259  
"stmt\_value\_index - Value index" on page 1259  
"stmt\_value\_isnull - Value has null value monitor element" on page 1260  
"stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261  
"stmt\_value\_type - Value type monitor element" on page 1261  
"uow\_id - Unit of work ID monitor element" on page 1401  
"member - Database member monitor element" on page 957

### **event\_bufferpool logical data group**

"bp\_id - Buffer pool identifier monitor element" on page 693  
"bp\_name - Buffer pool name monitor element" on page 693  
"db\_name - Database name monitor element" on page 773  
"db\_path - Database Path" on page 774  
"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_read\_time - Direct read time monitor element" on page 792  
"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_write\_reqs - Direct write requests monitor element" on page 796

"direct\_write\_time - Direct write time monitor element" on page 798

"direct\_writes - Direct writes to database monitor element" on page 800

"event\_time - Event Time" on page 811

"evmon\_activates - Number of event monitor activations" on page 813

"evmon\_flushes - Number of Event Monitor Flushes" on page 819

"files\_closed - Database files closed monitor element" on page 848

"partial\_record - Partial Record monitor element" on page 1014

"pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element" on page 1026

"pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element" on page 1027

"pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element" on page 1028

"pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element" on page 1032

"pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element" on page 1033

"pool\_async\_read\_time - Buffer Pool Asynchronous Read Time" on page 1033

"pool\_async\_write\_time - Buffer pool asynchronous write time monitor element" on page 1034

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113

"pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128

"block\_ios - Number of block I/O requests monitor element" on page 690

"pages\_from\_block\_ios - Total number of pages read by block I/O monitor element" on page 1011

"pages\_from\_vectored\_ios - Total number of pages read by vectored I/O monitor element" on page 1012

"pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element" on page 1031

"pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element" on page 1038

"pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element" on page 1038

"pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element" on page 1039

"pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element" on page 1087  
"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136  
"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139  
"pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141  
"unread\_prefetch\_pages - Unread prefetch pages monitor element" on page 1399  
"vectored\_ios - Number of vectored I/O requests monitor element" on page 1419

### **event\_conn logical data group**

"acc\_curs\_blk - Accepted Block Cursor Requests" on page 636  
"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_priority - Application Agent Priority" on page 671  
"appl\_priority\_type - Application Priority Type" on page 672  
"appl\_section\_inserts - Section Inserts monitor element" on page 672  
"appl\_section\_lookups - Section Lookups" on page 673  
"authority\_bitmap - User authorization level monitor element" on page 687  
"authority\_lvl - User authorization level monitor element" on page 688  
"binds\_precompiles - Binds/Precompiles Attempted" on page 690  
"cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698  
"cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699  
"cat\_cache\_overflows - Catalog Cache Overflows" on page 700  
"commit\_sql\_stmts - Commit Statements Attempted" on page 721  
"ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements" on page 781  
"deadlocks - Deadlocks detected monitor element" on page 783  
"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_read\_time - Direct read time monitor element" on page 792  
"direct\_reads - Direct reads from database monitor element" on page 794  
"direct\_write\_reqs - Direct write requests monitor element" on page 796  
"direct\_write\_time - Direct write time monitor element" on page 798  
"direct\_writes - Direct writes to database monitor element" on page 800  
"disconn\_time - Database Deactivation Timestamp" on page 803  
"dynamic\_sql\_stmts - Dynamic SQL Statements Attempted" on page 804  
"failed\_sql\_stmts - Failed Statement Operations" on page 821  
"hash\_join\_overflows - Hash Join Overflows" on page 867  
"hash\_join\_small\_overflows - Hash Join Small Overflows" on page 867  
"int\_auto\_rebinds - Internal Automatic Rebinds" on page 879  
"int\_commits - Internal commits monitor element" on page 880  
"int\_deadlock\_rollback - Internal Rollbacks Due To Deadlock" on page 881  
"int\_rollback - Internal rollbacks monitor element" on page 882  
"int\_rows\_deleted - Internal Rows Deleted" on page 884  
"int\_rows\_inserted - Internal Rows Inserted" on page 884

“int\_rows\_updated - Internal Rows Updated” on page 885

“lock\_escalation - Lock escalation monitor element” on page 904

“lock\_timeouts - Number of lock timeouts monitor element” on page 920

“lock\_wait\_time - Time waited on locks monitor element” on page 924

“lock\_waits - Lock waits monitor element” on page 928

“olap\_func\_overflows - OLAP Function Overflows monitor element” on page 997

“partial\_record - Partial Record monitor element” on page 1014

“int\_rows\_updated - Internal Rows Updated” on page 885

“pkg\_cache\_inserts - Package cache inserts monitor element” on page 1020

“pkg\_cache\_lookups - Package cache lookups monitor element” on page 1021

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151

“priv\_workspace\_num\_overflows - Private Workspace Overflows” on page 1156

“priv\_workspace\_section\_inserts - Private Workspace Section Inserts” on page 1157

“priv\_workspace\_section\_lookups - Private Workspace Section Lookups” on page 1158

“priv\_workspace\_size\_top - Maximum Private Workspace Size” on page 1158

“rej\_curs\_blk - Rejected Block Cursor Requests” on page 1173

“rollback\_sql\_stmts - Rollback Statements Attempted” on page 1184

“rows\_read - Rows read monitor element” on page 1191

“rows\_selected - Rows Selected” on page 1195

“rows\_written - Rows Written” on page 1196

“select\_sql\_stmts - Select SQL Statements Executed” on page 1203

“sequence\_no - Sequence number monitor element” on page 1204

“shr\_workspace\_num\_overflows - Shared Workspace Overflows” on page 1211  
 “shr\_workspace\_section\_inserts - Shared Workspace Section Inserts” on page 1212  
 “shr\_workspace\_section\_lookups - Shared Workspace Section Lookups” on page 1213  
 “shr\_workspace\_size\_top - Maximum Shared Workspace Size” on page 1213  
 “sort\_overflows - Sort overflows monitor element” on page 1224  
 “static\_sql\_stmts - Static SQL Statements Attempted” on page 1241  
 “system\_cpu\_time - System CPU time monitor element” on page 1267  
 “total\_hash\_joins - Total Hash Joins” on page 1338  
 “total\_hash\_loops - Total Hash Loops” on page 1338  
 “total\_olap\_funcs - Total OLAP Functions monitor element” on page 1346  
 “total\_sec\_cons - Secondary Connections” on page 1366  
 “total\_sort\_time - Total sort time monitor element” on page 1374  
 “total\_sorts - Total sorts monitor element” on page 1375  
 “uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed” on page 1398  
 “unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399  
 “user\_cpu\_time - User CPU time monitor element” on page 1411  
 “x\_lock\_escals - Exclusive lock escalations monitor element” on page 1429  
 “xquery\_stmts - XQuery Statements Attempted” on page 1431  
 “appl\_status - Application status monitor element” on page 673  
 “cat\_cache\_size\_top - Catalog cache high watermark monitor element” on page 701  
 “coord\_node - Coordinating Node” on page 750  
 “elapsed\_exec\_time - Statement Execution Elapsed Time” on page 806  
 “evmon\_flushes - Number of Event Monitor Flushes” on page 819  
 “lock\_escals - Number of lock escalations monitor element” on page 905  
 “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123  
 “pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125  
 “pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136  
 “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139  
 “pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141  
 “rows\_deleted - Rows deleted monitor element” on page 1188  
 “rows\_inserted - Rows inserted monitor element” on page 1189  
 “rows\_updated - Rows updated monitor element” on page 1196  
 CAT\_CACHE\_HEAP\_FULL

### **event\_connheader logical data group**

“agent\_id - Application handle (agent ID) monitor element” on page 650  
 “appl\_id - Application ID monitor element” on page 666  
 “appl\_name - Application name monitor element” on page 670  
 “auth\_id - Authorization ID” on page 686

"client\_db\_alias - Database Alias Used by Application" on page 711  
"client\_pid - Client process ID monitor element" on page 713  
"client\_platform - Client operating platform monitor element" on page 714  
"client\_prdid - Client product and version ID monitor element" on page 715  
"client\_protocol - Client communication protocol monitor element" on page 716  
"codepage\_id - ID of Code Page Used by Application" on page 718  
"conn\_time - Time of database connection monitor element" on page 736  
"corr\_token - DRDA Correlation Token" on page 751  
"execution\_id - User Login ID" on page 820  
"node\_number - Node Number" on page 970  
"sequence\_no - Sequence number monitor element" on page 1204  
"territory\_code - Database Territory Code" on page 1303  
"client\_nname - Client name monitor element" on page 713

### **event\_connmemuse logical data group**

"node\_number - Node Number" on page 970  
"pool\_config\_size - Configured Size of Memory Pool" on page 1040  
"pool\_cur\_size - Current Size of Memory Pool" on page 1041  
"pool\_id - Memory Pool Identifier" on page 1071  
"pool\_secondary\_id - Memory Pool Secondary Identifier" on page 1114  
"pool\_watermark - Memory Pool Watermark" on page 1127  
"appl\_id - Application ID monitor element" on page 666  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819  
POOL\_LIST\_ID  
POOL\_MAX\_SIZE

### **event\_data\_value logical data group**

"deadlock\_id - Deadlock Event Identifier" on page 782  
"deadlock\_node - Partition Number Where Deadlock Occurred" on page 783  
"evmon\_activates - Number of event monitor activations" on page 813  
"participant\_no - Participant within Deadlock" on page 1015  
"stmt\_history\_id - Statement history identifier" on page 1246  
"stmt\_value\_data - Value data" on page 1259  
"stmt\_value\_index - Value index" on page 1259  
"stmt\_value\_isnull - Value has null value monitor element" on page 1260  
"stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261  
"stmt\_value\_type - Value type monitor element" on page 1261

### **event\_db logical data group**

"active\_hash\_joins - Active hash joins" on page 644  
"appl\_section\_inserts - Section Inserts monitor element" on page 672  
"appl\_section\_lookups - Section Lookups" on page 673  
"async\_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 678  
"binds\_precompiles - Binds/Precompiles Attempted" on page 690

"blocks\_pending\_cleanup - Pending cleanup rolled-out blocks monitor element" on page 692

"cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698

"cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699

"cat\_cache\_overflows - Catalog Cache Overflows" on page 700

"cat\_cache\_size\_top - Catalog cache high watermark monitor element" on page 701

"catalog\_node - Catalog Node Number" on page 702

"catalog\_node\_name - Catalog Node Network Name" on page 702

"commit\_sql\_stmts - Commit Statements Attempted" on page 721

"connections\_top - Maximum Number of Concurrent Connections" on page 737

"db\_heap\_top - Maximum Database Heap Allocated" on page 773

"ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements" on page 781

"deadlocks - Deadlocks detected monitor element" on page 783

"direct\_read\_reqs - Direct read requests monitor element" on page 790

"direct\_read\_time - Direct read time monitor element" on page 792

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_write\_reqs - Direct write requests monitor element" on page 796

"direct\_write\_time - Direct write time monitor element" on page 798

"direct\_writes - Direct writes to database monitor element" on page 800

"disconn\_time - Database Deactivation Timestamp" on page 803

"dynamic\_sql\_stmts - Dynamic SQL Statements Attempted" on page 804

"evmon\_activates - Number of event monitor activations" on page 813

"evmon\_flushes - Number of Event Monitor Flushes" on page 819

"failed\_sql\_stmts - Failed Statement Operations" on page 821

"files\_closed - Database files closed monitor element" on page 848

"hash\_join\_overflows - Hash Join Overflows" on page 867

"hash\_join\_small\_overflows - Hash Join Small Overflows" on page 867

"int\_auto\_rebinds - Internal Automatic Rebinds" on page 879

"int\_commits - Internal commits monitor element" on page 880

"int\_rollback - Internal rollbacks monitor element" on page 882

"int\_rows\_deleted - Internal Rows Deleted" on page 884

"int\_rows\_inserted - Internal Rows Inserted" on page 884

"int\_rows\_updated - Internal Rows Updated" on page 885

"lock\_escals - Number of lock escalations monitor element" on page 905

"lock\_timeouts - Number of lock timeouts monitor element" on page 920

"lock\_wait\_time - Time waited on locks monitor element" on page 924

"lock\_waits - Lock waits monitor element" on page 928

"log\_held\_by\_dirty\_pages - Amount of Log Space Accounted for by Dirty Pages" on page 937

"log\_read\_time - Log Read Time" on page 938

"log\_reads - Number of Log Pages Read" on page 938

"log\_to\_redo\_for\_recovery - Amount of Log to be Redone for Recovery" on page 939

"log\_write\_time - Log Write Time" on page 940

"log\_writes - Number of Log Pages Written" on page 940

“num\_log\_read\_io - Number of Log Reads” on page 978

“num\_log\_write\_io - Number of Log Writes” on page 978

“num\_threshold\_violations - Number of threshold violations monitor element” on page 981

“olap\_func\_overflows - OLAP Function Overflows monitor element” on page 997

“partial\_record - Partial Record monitor element” on page 1014

“pkg\_cache\_inserts - Package cache inserts monitor element” on page 1020

“pkg\_cache\_lookups - Package cache lookups monitor element” on page 1021

“pkg\_cache\_num\_overflows - Package Cache Overflows” on page 1023

“pkg\_cache\_size\_top - Package cache high watermark” on page 1023

“pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element” on page 1026

“pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element” on page 1027

“pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element” on page 1028

“pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element” on page 1031

“pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element” on page 1032

“pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element” on page 1033

“pool\_async\_read\_time - Buffer Pool Asynchronous Read Time” on page 1033

“pool\_async\_write\_time - Buffer pool asynchronous write time monitor element” on page 1034

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_drty\_pg\_steal\_clns - Buffer pool victim page cleaners triggered monitor element” on page 1055

“pool\_drty\_pg\_thrsh\_clns - Buffer pool threshold cleaners triggered monitor element” on page 1056

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_lsn\_gap\_clns - Buffer pool log space cleaners triggered monitor element” on page 1086

“pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element” on page 1087

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121

"pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128

"post\_shrthreshold\_hash\_joins - Post threshold hash joins" on page 1143

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"prefetch\_wait\_time - Time waited for prefetch monitor element" on page 1151

"priv\_workspace\_num\_overflows - Private Workspace Overflows" on page 1156

"priv\_workspace\_section\_inserts - Private Workspace Section Inserts" on page 1157

"priv\_workspace\_section\_lookups - Private Workspace Section Lookups" on page 1158

"priv\_workspace\_size\_top - Maximum Private Workspace Size" on page 1158

"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184

"rows\_deleted - Rows deleted monitor element" on page 1188

"rows\_inserted - Rows inserted monitor element" on page 1189

"rows\_read - Rows read monitor element" on page 1191

"rows\_selected - Rows Selected" on page 1195

"rows\_updated - Rows updated monitor element" on page 1196

"sec\_log\_used\_top - Maximum Secondary Log Space Used" on page 1199

"select\_sql\_stmts - Select SQL Statements Executed" on page 1203

"server\_platform - Server Operating System" on page 1206

"shr\_workspace\_num\_overflows - Shared Workspace Overflows" on page 1211

"shr\_workspace\_section\_inserts - Shared Workspace Section Inserts" on page 1212

"shr\_workspace\_section\_lookups - Shared Workspace Section Lookups" on page 1213

"shr\_workspace\_size\_top - Maximum Shared Workspace Size" on page 1213

"sort\_overflows - Sort overflows monitor element" on page 1224

"static\_sql\_stmts - Static SQL Statements Attempted" on page 1241

"stats\_cache\_size - Size of statistics cache monitor element" on page 1242

"stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element" on page 1243

"stats\_fabrications - Total number of statistics fabrications monitor elements" on page 1244

"sync\_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 1265

"sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 1266

"tot\_log\_used\_top - Maximum Total Log Space Used" on page 1310

"total\_cons - Connects Since Database Activation" on page 1324

"total\_hash\_joins - Total Hash Joins" on page 1338

“total\_hash\_loops - Total Hash Loops” on page 1338  
“total\_olap\_funcs - Total OLAP Functions monitor element” on page 1346  
“total\_sort\_time - Total sort time monitor element” on page 1374  
“total\_sorts - Total sorts monitor element” on page 1375  
“uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed” on page 1398  
“unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399  
“x\_lock\_escals - Exclusive lock escalations monitor element” on page 1429  
“xquery\_stmts - XQuery Statements Attempted” on page 1431  
“elapsed\_exec\_time - Statement Execution Elapsed Time” on page 806  
“num\_log\_part\_page\_io - Number of Partial Log Page Writes” on page 977  
“pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element” on page 1038  
“pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element” on page 1038  
“pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element” on page 1039  
“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123  
“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125  
“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136  
“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139  
“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141  
“sort\_shrheap\_top - Sort share heap high watermark” on page 1226  
CAT\_CACHE\_HEAP\_FULL  
LOG\_FILE\_ARCHIVE  
LOG\_FILE\_NUM\_CURR  
LOG\_FILE\_NUM\_FIRST  
LOG\_FILE\_NUM\_LAST  
NUM\_LOG\_BUFF\_FULL  
NUM\_LOG\_DATA\_IN\_BUFF

### **event\_dbheader logical data group**

“conn\_time - Time of database connection monitor element” on page 736  
“db\_name - Database name monitor element” on page 773  
“db\_path - Database Path” on page 774

### **event\_dbmemuse logical data group**

“node\_number - Node Number” on page 970  
“pool\_config\_size - Configured Size of Memory Pool” on page 1040  
“pool\_cur\_size - Current Size of Memory Pool” on page 1041  
“pool\_id - Memory Pool Identifier” on page 1071  
“pool\_watermark - Memory Pool Watermark” on page 1127  
“evmon\_activates - Number of event monitor activations” on page 813

“evmon\_flushes - Number of Event Monitor Flushes” on page 819  
“pool\_secondary\_id - Memory Pool Secondary Identifier” on page 1114  
POOL\_MAX\_SIZE

### **event\_deadlock\_logical\_data\_group**

“deadlock\_id - Deadlock Event Identifier” on page 782  
“deadlock\_node - Partition Number Where Deadlock Occurred” on page 783  
“dl\_conns - Connections involved in deadlock monitor element” on page 804  
“evmon\_activates - Number of event monitor activations” on page 813  
“rolled\_back\_agent\_id - Rolled Back Agent” on page 1185  
“rolled\_back\_appl\_id - Rolled Back Application” on page 1186  
“rolled\_back\_participant\_no - Rolled back application participant monitor element” on page 1186  
“rolled\_back\_sequence\_no - Rolled Back Sequence Number” on page 1186  
“start\_time - Event Start Time” on page 1241

### **event\_detailed\_dlconn\_logical\_data\_group**

“agent\_id - Application handle (agent ID) monitor element” on page 650  
“appl\_id - Application ID monitor element” on page 666  
“appl\_id\_holding\_lk - Application ID Holding Lock” on page 668  
“blocking\_cursor - Blocking Cursor” on page 691  
“consistency\_token - Package consistency token monitor element” on page 738  
“creator - Application Creator” on page 763  
“cursor\_name - Cursor Name” on page 766  
“data\_partition\_id - Data partition identifier monitor element” on page 767  
“deadlock\_id - Deadlock Event Identifier” on page 782  
“deadlock\_node - Partition Number Where Deadlock Occurred” on page 783  
“evmon\_activates - Number of event monitor activations” on page 813  
“lock\_escalation - Lock escalation monitor element” on page 904  
“lock\_mode - Lock mode monitor element” on page 912  
“lock\_mode\_requested - Lock mode requested monitor element” on page 913  
“lock\_node - Lock Node” on page 915  
“lock\_object\_name - Lock Object Name” on page 915  
“lock\_object\_type - Lock object type waited on monitor element” on page 916  
“lock\_wait\_start\_time - Lock wait start timestamp monitor element” on page 923  
“locks\_held - Locks held monitor element” on page 932  
“locks\_in\_list - Number of Locks Reported” on page 933  
“package\_name - Package name monitor element” on page 1006  
“package\_version\_id - Package version monitor element” on page 1007  
“participant\_no - Participant within Deadlock” on page 1015  
“participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application” on page 1015  
“section\_number - Section number monitor element” on page 1201  
“sequence\_no - Sequence number monitor element” on page 1204  
“sequence\_no\_holding\_lk - Sequence Number Holding Lock” on page 1204

“start\_time - Event Start Time” on page 1241  
“stmt\_operation/operation - Statement operation monitor element” on page 1250  
“stmt\_text - SQL statement text monitor element” on page 1255  
“stmt\_type - Statement type monitor element” on page 1256  
“table\_name - Table name monitor element” on page 1269  
“table\_schema - Table schema name monitor element” on page 1271  
“tablespace\_name - Table space name monitor element” on page 1279  
“lock\_attributes - Lock attributes monitor element” on page 901  
“lock\_count - Lock count monitor element” on page 902  
“lock\_current\_mode - Original lock mode before conversion monitor element” on page 903  
“lock\_hold\_count - Lock hold count monitor element” on page 911  
“lock\_name - Lock name monitor element” on page 914  
“lock\_release\_flags - Lock release flags monitor element” on page 918  
“tpmon\_acc\_str - TP monitor client accounting string monitor element” on page 1387  
“tpmon\_client\_app - TP monitor client application name monitor element” on page 1388  
“tpmon\_client\_userid - TP monitor client user ID monitor element” on page 1388  
“tpmon\_client\_wkstn - TP monitor client workstation name monitor element” on page 1389

### **event\_dlconn logical data group**

“agent\_id - Application handle (agent ID) monitor element” on page 650  
“appl\_id - Application ID monitor element” on page 666  
“appl\_id\_holding\_lk - Application ID Holding Lock” on page 668  
“data\_partition\_id - Data partition identifier monitor element” on page 767  
“deadlock\_id - Deadlock Event Identifier” on page 782  
“deadlock\_node - Partition Number Where Deadlock Occurred” on page 783  
“evmon\_activates - Number of event monitor activations” on page 813  
“lock\_attributes - Lock attributes monitor element” on page 901  
“lock\_count - Lock count monitor element” on page 902  
“lock\_current\_mode - Original lock mode before conversion monitor element” on page 903  
“lock\_escalation - Lock escalation monitor element” on page 904  
“lock\_hold\_count - Lock hold count monitor element” on page 911  
“lock\_mode - Lock mode monitor element” on page 912  
“lock\_mode\_requested - Lock mode requested monitor element” on page 913  
“lock\_name - Lock name monitor element” on page 914  
“lock\_node - Lock Node” on page 915  
“lock\_object\_name - Lock Object Name” on page 915  
“lock\_object\_type - Lock object type waited on monitor element” on page 916  
“lock\_release\_flags - Lock release flags monitor element” on page 918  
“lock\_wait\_start\_time - Lock wait start timestamp monitor element” on page 923

"participant\_no - Participant within Deadlock" on page 1015  
"participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application" on page 1015  
"sequence\_no - Sequence number monitor element" on page 1204  
"sequence\_no\_holding\_lk - Sequence Number Holding Lock" on page 1204  
"start\_time - Event Start Time" on page 1241  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"tablespace\_name - Table space name monitor element" on page 1279  
"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389

### **event\_histogrambin logical data group**

"bin\_id - Histogram bin identifier monitor element" on page 690  
"bottom - Histogram bin bottom monitor element" on page 692  
"histogram\_type - Histogram type monitor element" on page 868  
"number\_in\_bin - Number in bin monitor element" on page 983  
"service\_class\_id - Service class ID monitor element" on page 1207  
"statistics\_timestamp - Statistics timestamp monitor element" on page 1241  
"top - Histogram bin top monitor element" on page 1310  
"work\_action\_set\_id - Work action set ID monitor element" on page 1424  
"work\_class\_id - Work class ID monitor element" on page 1425  
"workload\_id - Workload ID monitor element" on page 1426  
"mon\_interval\_id - Monitor interval identifier monitor element" on page 967  
"member - Database member monitor element" on page 957

### **event\_log\_header logical data group**

"byte\_order - Byte Order of Event Data" on page 697  
"codepage\_id - ID of Code Page Used by Application" on page 718  
"event\_monitor\_name - Event Monitor Name" on page 811  
"num\_nodes\_in\_db2\_instance - Number of Nodes in Partition" on page 980  
"server\_instance\_name - Server Instance Name" on page 1205  
"server\_prdid - Server Product/Version ID" on page 1206  
"territory\_code - Database Territory Code" on page 1303  
"version - Version of Monitor Data" on page 1419

### **event\_overflow logical data group**

"count - Number of Event Monitor Overflows" on page 752  
"first\_overflow\_time - Time of First Event Overflow" on page 849  
"last\_overflow\_time - Time of Last Event Overflow" on page 895  
"node\_number - Node Number" on page 970

## **event\_qstats logical data group**

- "last\_wlm\_reset - Time of last reset monitor element" on page 897*
- "queue\_assignments\_total - Queue assignments total monitor element" on page 1165*
- "queue\_size\_top - Queue size top monitor element" on page 1165*
- "queue\_time\_total - Queue time total monitor element" on page 1166*
- "service\_subclass\_name - Service subclass name monitor element" on page 1208*
- "service\_superclass\_name - Service superclass name monitor element" on page 1209*
- "statistics\_timestamp - Statistics timestamp monitor element" on page 1241*
- "threshold\_domain - Threshold domain monitor element" on page 1305*
- "threshold\_name - Threshold name monitor element" on page 1306*
- "threshold\_predicate - Threshold predicate monitor element" on page 1307*
- "thresholdid - Threshold ID monitor element" on page 1308*
- "work\_action\_set\_name - Work action set name monitor element" on page 1424*
- "work\_class\_name - Work class name monitor element" on page 1425*
- "mon\_interval\_id - Monitor interval identifier monitor element" on page 967*
- "member - Database member monitor element" on page 957*

## **event\_scstats logical data group**

- "act\_cpu\_time\_top - Activity CPU time top monitor element" on page 638*
- "act\_remapped\_in - Activities remapped in monitor element" on page 641*
- "act\_remapped\_out - Activities remapped out monitor element" on page 641*
- "act\_rows\_read\_top - Activity rows read top monitor element" on page 641*
- "act\_throughput - Activity throughput monitor element" on page 643*
- "agg\_temp\_tablespace\_top - Aggregate temporary table space top monitor element" on page 660*
- "concurrent\_act\_top - Concurrent activity top monitor element" on page 725*
- "concurrent\_wlo\_top - Concurrent workload occurrences top monitor element" on page 727*
- "concurrent\_connection\_top - Concurrent connection top monitor element" on page 726*
- "coord\_act\_aborted\_total - Coordinator activities aborted total monitor element" on page 741*
- "coord\_act\_completed\_total - Coordinator activities completed total monitor element" on page 742*
- "coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element" on page 742*
- "coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element" on page 743*
- "coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element" on page 744*
- "coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element" on page 745*
- "coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element" on page 746*
- "coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element" on page 747*

"coord\_act\_rejected\_total - Coordinator activities rejected total monitor element" on page 748

"cost\_estimate\_top - Cost estimate top monitor element" on page 752

"cpu\_utilization - CPU utilization monitor element" on page 759

details\_xml (This XML document is a metrics document of type system\_metrics, as described in the XML schema document sql11ib/misc/DB2MonCommon.xsd. Metrics reported for the default subclass SYSDEFAULTSUBCLASS under the superclass SYSDEFAULTSYSTEMCLASS have a value of 0.)

"last\_wlm\_reset - Time of last reset monitor element" on page 897

"request\_exec\_time\_avg - Request execution time average monitor element" on page 1183

"rows\_returned\_top - Actual rows returned top monitor element" on page 1194

"service\_class\_id - Service class ID monitor element" on page 1207

"service\_subclass\_name - Service subclass name monitor element" on page 1208

"service\_superclass\_name - Service superclass name monitor element" on page 1209

"statistics\_timestamp - Statistics timestamp monitor element" on page 1241

"temp\_tablespace\_top - Temporary table space top monitor element" on page 1303

"total\_cpu\_time - Total CPU time monitor element" on page 1330

"total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element" on page 1332

"uow\_completed\_total - Total completed units of work monitor element" on page 1400

"uow\_lifetime\_avg - Unit of work lifetime average monitor element" on page 1402

"uow\_throughput - Unit of work throughput monitor element" on page 1406

"uow\_total\_time\_top - UOW total time top monitor element" on page 1406

"app\_act\_aborted\_total - Total failed external coordinator activities monitor element" on page 661

"app\_act\_completed\_total - Total successful external coordinator activities monitor element" on page 662

"app\_act\_rejected\_total - Total rejected external coordinator activities monitor element" on page 664

"mon\_interval\_id - Monitor interval identifier monitor element" on page 967

"member - Database member monitor element" on page 957

### **event\_start logical data group**

"start\_time - Event Start Time" on page 1241

### **event\_stmt logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650

"agents\_top - Number of Agents Created" on page 658

"appl\_id - Application ID monitor element" on page 666

"blocking\_cursor - Blocking Cursor" on page 691

"consistency\_token - Package consistency token monitor element" on page 738

"creator - Application Creator" on page 763

"cursor\_name - Cursor Name" on page 766

“fetch\_count - Number of Successful Fetches” on page 847  
 “int\_rows\_deleted - Internal Rows Deleted” on page 884  
 “int\_rows\_inserted - Internal Rows Inserted” on page 884  
 “int\_rows\_updated - Internal Rows Updated” on page 885  
 “package\_name - Package name monitor element” on page 1006  
 “package\_version\_id - Package version monitor element” on page 1007  
 “partial\_record - Partial Record monitor element” on page 1014  
 “pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048  
 “pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050  
 “pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080  
 “pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082  
 “pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116  
 “pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118  
 “pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120  
 “pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121  
 “rows\_read - Rows read monitor element” on page 1191  
 “rows\_written - Rows Written” on page 1196  
 “section\_number - Section number monitor element” on page 1201  
 “sequence\_no - Sequence number monitor element” on page 1204  
 “sort\_overflows - Sort overflows monitor element” on page 1224  
 “sql\_req\_id - Request Identifier for SQL Statement” on page 1228  
 “sqlca - SQL Communications Area (SQLCA)” on page 1229  
 “start\_time - Event Start Time” on page 1241  
 “stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element” on page 1243  
 “stmt\_operation/operation - Statement operation monitor element” on page 1250  
 “stmt\_text - SQL statement text monitor element” on page 1255  
 “stmt\_type - Statement type monitor element” on page 1256  
 “stop\_time - Event Stop Time” on page 1262  
 “sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element” on page 1266  
 “system\_cpu\_time - System CPU time monitor element” on page 1267  
 “total\_sort\_time - Total sort time monitor element” on page 1374  
 “total\_sorts - Total sorts monitor element” on page 1375  
 “user\_cpu\_time - User CPU time monitor element” on page 1411  
 “evmon\_flushes - Number of Event Monitor Flushes” on page 819  
 “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125

"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136

"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139

### **event\_stmt\_history logical data group**

"comp\_env\_desc - Compilation environment monitor element" on page 722

"creator - Application Creator" on page 763

"deadlock\_id - Deadlock Event Identifier" on page 782

"deadlock\_node - Partition Number Where Deadlock Occurred" on page 783

"evmon\_activates - Number of event monitor activations" on page 813

"package\_name - Package name monitor element" on page 1006

"package\_version\_id - Package version monitor element" on page 1007

"participant\_no - Participant within Deadlock" on page 1015

"section\_number - Section number monitor element" on page 1201

"sequence\_no - Sequence number monitor element" on page 1204

"stmt\_first\_use\_time - Statement first use timestamp monitor element" on page 1246

"stmt\_history\_id - Statement history identifier" on page 1246

"stmt\_invocation\_id - Statement invocation identifier monitor element" on page 1247

"stmt\_isolation - Statement isolation" on page 1248

"stmt\_last\_use\_time - Statement last use timestamp monitor element" on page 1248

"stmt\_lock\_timeout - Statement lock timeout monitor element" on page 1249

"stmt\_nest\_level - Statement nesting level monitor element" on page 1249

"stmt\_pkgcache\_id - Statement package cache identifier monitor element" on page 1251

"stmt\_query\_id - Statement query identifier monitor element" on page 1252

"stmt\_source\_id - Statement source identifier" on page 1253

"stmt\_text - SQL statement text monitor element" on page 1255

"stmt\_type - Statement type monitor element" on page 1256

"appl\_id - Application ID monitor element" on page 666

### **event\_subsection logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650

"num\_agents - Number of Agents Working on a Statement" on page 971

"partial\_record - Partial Record monitor element" on page 1014

"ss\_exec\_time - Subsection Execution Elapsed Time" on page 1237

"ss\_node\_number - Subsection Node Number" on page 1238

"ss\_number - Subsection number monitor element" on page 1238

"ss\_sys\_cpu\_time - System CPU Time used by Subsection" on page 1239

"ss\_usr\_cpu\_time - User CPU Time used by Subsection" on page 1239

"tq\_max\_send\_spills - Maximum number of table queue buffers overflows" on page 1390

"tq\_rows\_read - Number of Rows Read from table queues" on page 1391  
"tq\_rows\_written - Number of rows written to table queues" on page 1391  
"tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396  
"appl\_id - Application ID monitor element" on page 666  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819  
"sql\_req\_id - Request Identifier for SQL Statement" on page 1228

### **event\_table logical data group**

"data\_object\_pages - Data Object Pages" on page 766  
"data\_partition\_id - Data partition identifier monitor element" on page 767  
"event\_time - Event Time" on page 811  
"evmon\_activates - Number of event monitor activations" on page 813  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819  
"index\_object\_pages - Index Object Pages" on page 875  
"lob\_object\_pages - LOB Object Pages" on page 898  
"long\_object\_pages - Long Object Pages" on page 941  
"overflow\_accesses - Accesses to overflowed records monitor element" on page 1004  
"page\_reorgs - Page reorganizations monitor element" on page 1009  
"partial\_record - Partial Record monitor element" on page 1014  
"rows\_read - Rows read monitor element" on page 1191  
"rows\_written - Rows Written" on page 1196  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"table\_type - Table type monitor element" on page 1273  
"tablespace\_id - Table space identification monitor element" on page 1276  
"xda\_object\_pages - XDA Object Pages" on page 1430

### **event\_tablespace logical data group**

"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_read\_time - Direct read time monitor element" on page 792  
"direct\_reads - Direct reads from database monitor element" on page 794  
"direct\_write\_reqs - Direct write requests monitor element" on page 796  
"direct\_write\_time - Direct write time monitor element" on page 798  
"direct\_writes - Direct writes to database monitor element" on page 800  
"event\_time - Event Time" on page 811  
"evmon\_activates - Number of event monitor activations" on page 813  
"evmon\_flushes - Number of Event Monitor Flushes" on page 819  
"files\_closed - Database files closed monitor element" on page 848  
"partial\_record - Partial Record monitor element" on page 1014  
"pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element" on page 1026  
"pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element" on page 1027  
"pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element" on page 1028

“pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element” on page 1031

“pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element” on page 1032

“pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element” on page 1033

“pool\_async\_read\_time - Buffer Pool Asynchronous Read Time” on page 1033

“pool\_async\_write\_time - Buffer pool asynchronous write time monitor element” on page 1034

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element” on page 1087

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“tablespace\_name - Table space name monitor element” on page 1279

“pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element” on page 1038

“pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element” on page 1038

“pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element” on page 1039

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

TABLESPACE\_FS\_CACHING

“unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399

## **event\_thresholdviolations logical data group**

- “activate\_timestamp - Activate timestamp monitor element” on page 644
- “activity\_collected - Activity collected monitor element” on page 645
- “activity\_id - Activity ID monitor element” on page 645
- “agent\_id - Application handle (agent ID) monitor element” on page 650
- “appl\_id - Application ID monitor element” on page 666
- “coord\_partition\_num - Coordinator partition number monitor element” on page 750
- “destination\_service\_class\_id - Destination service class ID monitor element” on page 787
- “source\_service\_class\_id - Source service class ID monitor element” on page 1227
- “threshold\_action - Threshold action monitor element” on page 1305
- “threshold\_maxvalue - Threshold maximum value monitor element” on page 1306
- “threshold\_predicate - Threshold predicate monitor element” on page 1307
- “threshold\_queuesize - Threshold queue size monitor element” on page 1308
- “thresholdid - Threshold ID monitor element” on page 1308
- “time\_of\_violation - Time of violation monitor element” on page 1309
- “uow\_id - Unit of work ID monitor element” on page 1401
- “member - Database member monitor element” on page 957

## **event\_wcstats logical data group**

- “act\_cpu\_time\_top - Activity CPU time top monitor element” on page 638
- “act\_rows\_read\_top - Activity rows read top monitor element” on page 641
- “act\_total - Activities total monitor element” on page 643
- “coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element” on page 742
- “coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element” on page 743
- “coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element” on page 744
- “coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element” on page 745
- “coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element” on page 746
- “coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element” on page 747
- “cost\_estimate\_top - Cost estimate top monitor element” on page 752
- “last\_wlm\_reset - Time of last reset monitor element” on page 897
- “rows\_returned\_top - Actual rows returned top monitor element” on page 1194
- “statistics\_timestamp - Statistics timestamp monitor element” on page 1241
- “temp\_tablespace\_top - Temporary table space top monitor element” on page 1303
- “work\_action\_set\_id - Work action set ID monitor element” on page 1424
- “work\_action\_set\_name - Work action set name monitor element” on page 1424
- “work\_class\_id - Work class ID monitor element” on page 1425

“work\_class\_name - Work class name monitor element” on page 1425  
“mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
“member - Database member monitor element” on page 957

### **event\_wlstats logical data group**

“act\_cpu\_time\_top - Activity CPU time top monitor element” on page 638  
“act\_rows\_read\_top - Activity rows read top monitor element” on page 641  
“act\_throughput - Activity throughput monitor element” on page 643  
“concurrent\_wlo\_act\_top - Concurrent WLO activity top monitor element” on page 726  
“concurrent\_wlo\_top - Concurrent workload occurrences top monitor element” on page 727  
“coord\_act\_aborted\_total - Coordinator activities aborted total monitor element” on page 741  
“coord\_act\_completed\_total - Coordinator activities completed total monitor element” on page 742  
“coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element” on page 742  
“coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element” on page 743  
“coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element” on page 744  
“coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element” on page 745  
“coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element” on page 746  
“coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element” on page 747  
“coord\_act\_rejected\_total - Coordinator activities rejected total monitor element” on page 748  
“cost\_estimate\_top - Cost estimate top monitor element” on page 752  
“cpu\_utilization - CPU utilization monitor element” on page 759  
details\_xml (This XML document is a metrics document of type system\_metrics, as described in the XML schema document sql11ib/misc/DB2MonCommon.xsd.)  
“last\_wlm\_reset - Time of last reset monitor element” on page 897  
“lock\_wait\_time\_top - Lock wait time top monitor element” on page 928  
“rows\_returned\_top - Actual rows returned top monitor element” on page 1194  
“statistics\_timestamp - Statistics timestamp monitor element” on page 1241  
“temp\_tablespace\_top - Temporary table space top monitor element” on page 1303  
“total\_cpu\_time - Total CPU time monitor element” on page 1330  
“total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332  
“uow\_completed\_total - Total completed units of work monitor element” on page 1400  
“uow\_lifetime\_avg - Unit of work lifetime average monitor element” on page 1402  
“uow\_throughput - Unit of work throughput monitor element” on page 1406

“uow\_total\_time\_top - UOW total time top monitor element” on page 1406  
“wlo\_completed\_total - Workload occurrences completed total monitor element” on page 1424  
“workload\_id - Workload ID monitor element” on page 1426  
“workload\_name - Workload name monitor element” on page 1427  
“app\_act\_aborted\_total - Total failed external coordinator activities monitor element” on page 661  
“app\_act\_completed\_total - Total successful external coordinator activities monitor element” on page 662  
“app\_act\_rejected\_total - Total rejected external coordinator activities monitor element” on page 664  
“lock\_wait\_time\_global\_top - Top global lock wait time monitor element” on page 928  
“mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
“member - Database member monitor element” on page 957

### **event\_xact logical data group**

“agent\_id - Application handle (agent ID) monitor element” on page 650  
“appl\_id - Application ID monitor element” on page 666  
“lock\_escals - Number of lock escalations monitor element” on page 905  
“lock\_wait\_time - Time waited on locks monitor element” on page 924  
“locks\_held\_top - Maximum number of locks held monitor element” on page 932  
“partial\_record - Partial Record monitor element” on page 1014  
“prev\_uow\_stop\_time - Previous Unit of Work Completion Timestamp” on page 1155  
“rows\_read - Rows read monitor element” on page 1191  
“rows\_written - Rows Written” on page 1196  
“sequence\_no - Sequence number monitor element” on page 1204  
“system\_cpu\_time - System CPU time monitor element” on page 1267  
“tpmon\_acc\_str - TP monitor client accounting string monitor element” on page 1387  
“tpmon\_client\_app - TP monitor client application name monitor element” on page 1388  
“tpmon\_client\_userid - TP monitor client user ID monitor element” on page 1388  
“tpmon\_client\_wkstn - TP monitor client workstation name monitor element” on page 1389  
“uow\_log\_space\_used - Unit of work log space used monitor element” on page 1403  
“uow\_start\_time - Unit of work start timestamp monitor element” on page 1404  
“uow\_status - Unit of Work Status” on page 1405  
“stop\_time - Event Stop Time” on page 1262  
“user\_cpu\_time - User CPU time monitor element” on page 1411  
“x\_lock\_escals - Exclusive lock escalations monitor element” on page 1429  
“evmon\_flushes - Number of Event Monitor Flushes” on page 819

## **evmonstart logical data group**

- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "member - Database member monitor element" on page 957*
- "event\_type - Event Type monitor element" on page 812*
- "db2start\_time - Start Database Manager Timestamp" on page 772*
- "db\_conn\_time - Database activation timestamp monitor element" on page 772*

## **lock logical data group**

- "partition\_key - Partitioning key monitor element" on page 1016*
- "dl\_conns - Connections involved in deadlock monitor element" on page 804*
- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "event\_type - Event Type monitor element" on page 812*
- "partition\_number - Partition Number" on page 1017*
- "rolled\_back\_participant\_no - Rolled back application participant monitor element" on page 1186*
- "deadlock\_type - Deadlock type monitor element" on page 783*

## **lock\_activity\_values logical data group**

- "partition\_key - Partitioning key monitor element" on page 1016*
- "activity\_id - Activity ID monitor element" on page 645*
- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "event\_type - Event Type monitor element" on page 812*
- "participant\_no - Participant within Deadlock" on page 1015*
- "partition\_number - Partition Number" on page 1017*
- "stmt\_value\_index - Value index" on page 1259*
- "stmt\_value\_isnull - Value has null value monitor element" on page 1260*
- "stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261*
- "stmt\_value\_type - Value type monitor element" on page 1261*
- "uow\_id - Unit of work ID monitor element" on page 1401*
- "stmt\_value\_data - Value data" on page 1259*
- "event\_id - Event ID monitor element" on page 810*

## **lock\_participant\_activities logical data group**

- "partition\_key - Partitioning key monitor element" on page 1016*
- "activity\_id - Activity ID monitor element" on page 645*
- "activity\_type - Activity type monitor element" on page 647*
- "consistency\_token - Package consistency token monitor element" on page 738*
- "effective\_isolation - Effective isolation monitor element" on page 805*
- "effective\_query\_degree - Effective query degree monitor element" on page 806*
- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "event\_type - Event Type monitor element" on page 812*

“incremental\_bind - Incremental bind monitor element” on page 874  
“package\_name - Package name monitor element” on page 1006  
“package\_schema - Package schema monitor element” on page 1007  
“package\_version\_id - Package version monitor element” on page 1007  
“participant\_no - Participant within Deadlock” on page 1015  
“partition\_number - Partition Number” on page 1017  
“query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element” on page 1162  
“reopt - Reopt bind option monitor element” on page 1176  
“section\_number - Section number monitor element” on page 1201  
“stmt\_first\_use\_time - Statement first use timestamp monitor element” on page 1246  
“stmt\_invocation\_id - Statement invocation identifier monitor element” on page 1247  
“stmt\_last\_use\_time - Statement last use timestamp monitor element” on page 1248  
“stmt\_lock\_timeout - Statement lock timeout monitor element” on page 1249  
“stmt\_nest\_level - Statement nesting level monitor element” on page 1249  
“stmt\_pkgcache\_id - Statement package cache identifier monitor element” on page 1251  
“stmt\_query\_id - Statement query identifier monitor element” on page 1252  
“stmt\_source\_id - Statement source identifier” on page 1253  
“stmt\_type - Statement type monitor element” on page 1256  
“uow\_id - Unit of work ID monitor element” on page 1401  
“stmt\_text - SQL statement text monitor element” on page 1255  
“stmt\_unicode - Statement unicode flag monitor element” on page 1258  
“stmt\_operation/operation - Statement operation monitor element” on page 1250

### **lock\_participants logical data group**

“partition\_key - Partitioning key monitor element” on page 1016  
“agent\_status - DCS Application Agents” on page 652  
“agent\_id - Application handle (agent ID) monitor element” on page 650  
“appl\_id - Application ID monitor element” on page 666  
“appl\_name - Application name monitor element” on page 670  
“application\_handle - Application handle monitor element” on page 675  
“auth\_id - Authorization ID” on page 686  
“client\_acctng - Client accounting string monitor element” on page 709  
“client\_applname - Client application name monitor element” on page 710  
“client\_userid - Client user ID monitor element” on page 717  
“client\_wrkstname - Client workstation name monitor element” on page 717  
“coord\_agent\_tid - Coordinator agent engine dispatchable unit ID monitor element” on page 747  
“current\_request - Current operation request monitor element” on page 766  
“event\_id - Event ID monitor element” on page 810  
“event\_timestamp - Event timestamp monitor element” on page 811  
“event\_type - Event Type monitor element” on page 812

"lock\_attributes - Lock attributes monitor element" on page 901  
 "lock\_count - Lock count monitor element" on page 902  
 "lock\_current\_mode - Original lock mode before conversion monitor element" on page 903  
 "lock\_escalation - Lock escalation monitor element" on page 904  
 "lock\_hold\_count - Lock hold count monitor element" on page 911  
 "lock\_mode - Lock mode monitor element" on page 912  
 "lock\_mode\_requested - Lock mode requested monitor element" on page 913  
 "lock\_name - Lock name monitor element" on page 914  
 "lock\_object\_type - Lock object type waited on monitor element" on page 916  
 LOCK\_OBJECT\_TYPE\_ID  
 "lock\_release\_flags - Lock release flags monitor element" on page 918  
 LOCK\_RRIID  
 "lock\_status - Lock status monitor element" on page 919  
 "lock\_timeout\_val - Lock timeout value monitor element" on page 920  
 "lock\_wait\_end\_time - Lock wait end timestamp monitor element" on page 923  
 "lock\_wait\_start\_time - Lock wait start timestamp monitor element" on page 923  
 "lock\_wait\_val - Lock wait value monitor element" on page 928  
 "member - Database member monitor element" on page 957  
 "object\_requested - Requested object monitor element" on page 992  
 "participant\_no - Participant within Deadlock" on page 1015  
 "participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application" on page 1015  
 "participant\_type - Participant type monitor element" on page 1016  
 "past\_activities\_wrapped - Past activities list wrapped monitor element" on page 1018  
 "service\_class\_id - Service class ID monitor element" on page 1207  
 "service\_subclass\_name - Service subclass name monitor element" on page 1208  
 "table\_file\_id - Table file ID monitor element" on page 1268  
 "table\_name - Table name monitor element" on page 1269  
 "table\_schema - Table schema name monitor element" on page 1271  
 "thresholdid - Threshold ID monitor element" on page 1308  
 "threshold\_name - Threshold name monitor element" on page 1306  
 "workload\_id - Workload ID monitor element" on page 1426  
 "workload\_name - Workload name monitor element" on page 1427  
 "agent\_tid - Agent thread ID monitor element" on page 653  
 "appl\_action - Application action monitor element" on page 665  
 "deadlock\_member - Deadlock member monitor element" on page 782  
 "queue\_start\_time - Queue start timestamp monitor element" on page 1165  
 "queued\_agents - Queued threshold agents monitor element" on page 1166  
 "service\_superclass\_name - Service superclass name monitor element" on page 1209  
 "tablespace\_name - Table space name monitor element" on page 1279  
 "xid - Transaction ID" on page 1431  
 "utility\_invocation\_id - Utility invocation ID" on page 1413

## **pkgcache logical data group**

- "partition\_key - Partitioning key monitor element" on page 1016
- "comp\_env\_desc - Compilation environment monitor element" on page 722
- "effective\_isolation - Effective isolation monitor element" on page 805
- "event\_id - Event ID monitor element" on page 810
- "event\_timestamp - Event timestamp monitor element" on page 811
- "executable\_id - Executable ID monitor element" on page 818
- "insert\_timestamp - Insert timestamp monitor element" on page 878
- "last\_metrics\_update - Metrics last update timestamp monitor element" on page 895
- "member - Database member monitor element" on page 957
- "num\_coord\_exec - Number of executions by coordinator agent monitor element" on page 972
- "num\_coord\_exec\_with\_metrics - Number of executions by coordinator agent with metrics monitor element" on page 972
- "num\_exec\_with\_metrics - Number of executions with metrics collected monitor element" on page 974
- "num\_executions - Statement executions monitor element" on page 973
- "package\_name - Package name monitor element" on page 1006
- "package\_schema - Package schema monitor element" on page 1007
- "package\_version\_id - Package version monitor element" on page 1007
- "partition\_number - Partition Number" on page 1017
- "prep\_time - Preparation time monitor element" on page 1154
- "query\_cost\_estimate - Query cost estimate monitor element" on page 1163
- "query\_data\_tag\_list - Estimated query data tag list monitor element" on page 1164
- "routine\_id - Routine ID monitor element" on page 1187
- "section\_env - Section environment monitor element" on page 1201
- "section\_number - Section number monitor element" on page 1201
- "section\_type - Section type indicator monitor element" on page 1202
- "stmt\_pkgcache\_id - Statement package cache identifier monitor element" on page 1251
- "stmt\_type\_id - Statement type identifier monitor element" on page 1257
- "total\_stats\_fabrication\_time - Total statistics fabrication time monitor element" on page 1378
- "total\_stats\_fabrications - Total statistics fabrications monitor elements" on page 1379
- "total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element" on page 1382
- "total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements" on page 1380
- "stmt\_text - SQL statement text monitor element" on page 1255
- "max\_coord\_stmt\_exec\_time - Maximum coordinator statement execution time monitor element" on page 943
- "max\_coord\_stmt\_exec\_timestamp - Maximum coordinator statement execution timestamp monitor element" on page 945

## **pkgcache\_metrics logical data group**

“partition\_key - Partitioning key monitor element” on page 1016

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“partition\_number - Partition Number” on page 1017

“wlm\_queue\_time\_total - Workload manager total queue time monitor element” on page 1422

“wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element” on page 1421

“fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element” on page 840

“fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element” on page 824

“fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element” on page 844

“fcm\_message\_send\_wait\_time - FCM message send wait time monitor element” on page 828

“lock\_wait\_time - Time waited on locks monitor element” on page 924

“lock\_waits - Lock waits monitor element” on page 928

“direct\_read\_time - Direct read time monitor element” on page 792

“direct\_read\_reqs - Direct read requests monitor element” on page 790

“direct\_write\_time - Direct write time monitor element” on page 798

“direct\_write\_reqs - Direct write requests monitor element” on page 796

“log\_buffer\_wait\_time - Log buffer wait time monitor element” on page 933

“num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element” on page 975

“log\_disk\_wait\_time - Log disk wait time monitor element” on page 935

“log\_disk\_waits\_total - Total log disk waits monitor element” on page 936

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“audit\_file\_write\_wait\_time - Audit file write wait time monitor element” on page 680

“audit\_file\_writes\_total - Total audit files written monitor element” on page 682

“audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element” on page 683

“audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element” on page 685

“diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element” on page 788

“diaglog\_writes\_total - Total diagnostic log file writes monitor element” on page 789

“fcm\_send\_wait\_time - FCM send wait time monitor element” on page 836

“fcm\_rcv\_wait\_time - FCM received wait time monitor element” on page 832

“total\_act\_wait\_time - Total activity wait time monitor element” on page 1312

“total\_section\_sort\_proc\_time - Total section sort processing time monitor element” on page 1368

“total\_section\_sorts - Total section sorts monitor element” on page 1371

“total\_section\_sort\_time - Total section sort time monitor element” on page 1370

“total\_act\_time - Total activity time monitor element” on page 1311

“rows\_read - Rows read monitor element” on page 1191

“rows\_modified - Rows modified monitor element” on page 1190

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“total\_cpu\_time - Total CPU time monitor element” on page 1330

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“direct\_reads - Direct reads from database monitor element” on page 794

“direct\_writes - Direct writes to database monitor element” on page 800

“rows\_returned - Rows returned monitor element” on page 1193

“deadlocks - Deadlocks detected monitor element” on page 783

“lock\_timeouts - Number of lock timeouts monitor element” on page 920

“lock\_escals - Number of lock escalations monitor element” on page 905

“fcm\_sends\_total - FCM sends total monitor element” on page 838

“fcm\_recvs\_total - FCM receives total monitor element” on page 834

“fcm\_send\_volume - FCM send volume monitor element” on page 835

“fcm\_recv\_volume - FCM received volume monitor element” on page 831

“fcm\_message\_sends\_total - Total FCM message sends monitor element” on page 830

“fcm\_message\_recvs\_total - Total FCM message receives monitor element” on page 826

"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827

"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823

"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846

"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842

"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843

"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839

"tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396

"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"sort\_overflows - Sort overflows monitor element" on page 1224

"audit\_events\_total - Total audit events monitor element" on page 679

"total\_sorts - Total sorts monitor element" on page 1375

"stmt\_exec\_time - Statement execution time monitor element" on page 1245

"coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element" on page 751

"total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element" on page 1356

"total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements" on page 1356

"total\_section\_proc\_time - Total section processing time monitor element" on page 1367

"total\_app\_section\_executions - Total application section executions monitor element" on page 1317

"total\_section\_time - Total section time monitor element" on page 1373

"total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element" on page 1359

"total\_routine\_user\_code\_time - Total routine user code time monitor element" on page 1360

"total\_routine\_time - Total routine time monitor element" on page 1357

"thresh\_violations - Number of threshold violations monitor element" on page 1304

"num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element" on page 979

"total\_routine\_invocations - Total routine invocations monitor elements" on page 1355

"lock\_wait\_time\_global - Lock wait time global monitor element" on page 926

"lock\_waits\_global - Lock waits global monitor element" on page 930

"reclaim\_wait\_time - Reclaim wait time monitor element" on page 1170

"spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element" on page 1236

"lock\_timeouts\_global - Lock timeouts global monitor element" on page 922

“lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element” on page 910

“lock\_escals\_locklist - Number of locklist lock escalations monitor element” on page 909

“lock\_escals\_global - Number of global lock escalations monitor element” on page 907

“cf\_wait\_time - cluster caching facility wait time monitor element” on page 703

“cf\_waits - Number of cluster caching facility waits monitor element” on page 703

“pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element” on page 1044

“pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element” on page 1045

“pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element” on page 1047

“pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element” on page 1042

“pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element” on page 1075

“pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements” on page 1077

“pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element” on page 1078

“pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element” on page 1074

“pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element” on page 1132

“pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element” on page 1134

“pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element” on page 1138

“pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element” on page 1131

“evmon\_wait\_time - Event monitor wait time monitor element” on page 814

“evmon\_waits\_total - Event monitor total waits monitor element” on page 816

“total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335

“total\_extended\_latch\_waits - Total extended latch waits monitor element” on page 1336

“total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332

“pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element” on page 1089

“pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element” on page 1093

“pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element” on page 1111

“pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element” on page 1087

"pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element" on page 1091

"pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element" on page 1109

### **pkgcache\_stmt\_args logical data group**

"event\_id - Event ID monitor element" on page 810

"event\_timestamp - Event timestamp monitor element" on page 811

"stmt\_value\_index - Value index" on page 1259

"stmt\_value\_isreopt - Variable used for statement reoptimization monitor element" on page 1261

"stmt\_value\_isnull - Value has null value monitor element" on page 1260

"stmt\_value\_type - Value type monitor element" on page 1261

"stmt\_value\_data - Value data" on page 1259

"member - Database member monitor element" on page 957

### **regvar logical data group**

"event\_id - Event ID monitor element" on page 810

"event\_timestamp - Event timestamp monitor element" on page 811

"member - Database member monitor element" on page 957

"event\_type - Event Type monitor element" on page 812

"regvar\_name - Registry variable name" on page 1172

"regvar\_value - Registry variable value" on page 1173

"regvar\_old\_value - Registry variable old value" on page 1173

"regvar\_level - Registry variable level" on page 1172

"regvar\_collection\_type - Registry variable collection type" on page 1172

### **sqlca logical data group**

sqlcab

sqlcaid

sqlcode

sqlerrd

sqlerrmc

sqlerrml

sqlerrp

sqlstate

sqlwarn

### **txncompletion logical data group**

"event\_id - Event ID monitor element" on page 810

"event\_timestamp - Event timestamp monitor element" on page 811

"member - Database member monitor element" on page 957

"event\_type - Event Type monitor element" on page 812

"global\_transaction\_id - Global transaction identifier monitor element" on page 851

"local\_transaction\_id - Local transaction identifier monitor element" on page 900

“savepoint\_id - Savepoint ID” on page 1198  
“uow\_id - Unit of work ID monitor element” on page 1401  
“ddl\_classification - DDL classification” on page 780  
“txn\_completion\_status - Transaction completion status” on page 1398

## **uow logical data group**

“partition\_key - Partitioning key monitor element” on page 1016  
“application\_handle - Application handle monitor element” on page 675  
“appl\_id - Application ID monitor element” on page 666  
“appl\_name - Application name monitor element” on page 670  
“client\_acctng - Client accounting string monitor element” on page 709  
“client\_applname - Client application name monitor element” on page 710  
“client\_hostname - Client hostname monitor element” on page 711  
“client\_pid - Client process ID monitor element” on page 713  
“client\_port\_number - Client port number monitor element” on page 715  
“client\_prdid - Client product and version ID monitor element” on page 715  
“client\_userid - Client user ID monitor element” on page 717  
“client\_wrkstname - Client workstation name monitor element” on page 717  
“completion\_status - Completion status monitor element” on page 722  
“conn\_time - Time of database connection monitor element” on page 736  
“coord\_member - Coordinator member monitor element” on page 749  
“event\_id - Event ID monitor element” on page 810  
“event\_timestamp - Event timestamp monitor element” on page 811  
“executable\_list\_size - Size of executable list monitor element” on page 819  
“global\_transaction\_id - Global transaction identifier monitor element” on page 851  
“intra\_parallel\_state - Current state of intrapartition parallelism monitor element” on page 886  
“local\_transaction\_id - Local transaction identifier monitor element” on page 900  
“member - Database member monitor element” on page 957  
“db\_conn\_time - Database activation timestamp monitor element” on page 772  
“mon\_interval\_id - Monitor interval identifier monitor element” on page 967  
“package\_list\_exceeded - Package list exceeded monitor element” on page 1006  
“package\_list\_size - Size of package list monitor element” on page 1006  
service\_class\_id - Service class ID  
service\_subclass\_name - Service subclass name  
service\_superclass\_name - Service superclass name  
“session\_auth\_id - Session authorization ID monitor element” on page 1211  
start\_time - Event start time  
stop\_time - Event stop time  
“system\_auth\_id - System authorization identifier monitor element” on page 1267  
UOW\_CLIENT\_PLATFORM  
UOW\_CLIENT\_PROTOCOL  
uow\_id - Unit of work ID

"uow\_log\_space\_used - Unit of work log space used monitor element" on page 1403  
workload\_id - Workload ID  
workload\_name - Workload name  
workload\_occurrence\_id - Workload occurrence identifier  
"client\_platform - Client operating platform monitor element" on page 714  
"client\_protocol - Client communication protocol monitor element" on page 716  
"executable\_list\_truncated - Executable list truncated monitor element" on page 819  
"connection\_start\_time - Connection start time monitor element" on page 736

### **uow\_executable\_list logical data group**

"partition\_key - Partitioning key monitor element" on page 1016  
"appl\_id - Application ID monitor element" on page 666  
"executable\_id - Executable ID monitor element" on page 818  
"lock\_wait\_time - Time waited on locks monitor element" on page 924  
"lock\_waits - Lock waits monitor element" on page 928  
"num\_executions - Statement executions monitor element" on page 973  
"partition\_number - Partition Number" on page 1017  
"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144  
"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150  
"rows\_read - Rows read monitor element" on page 1191  
"sort\_overflows - Sort overflows monitor element" on page 1224  
"total\_act\_time - Total activity time monitor element" on page 1311  
"total\_act\_wait\_time - Total activity wait time monitor element" on page 1312  
"total\_cpu\_time - Total CPU time monitor element" on page 1330  
"total\_sorts - Total sorts monitor element" on page 1375  
uow\_id - Unit of work ID

### **uow\_metrics logical data group**

"partition\_key - Partitioning key monitor element" on page 1016  
"appl\_id - Application ID monitor element" on page 666  
"partition\_number - Partition Number" on page 1017  
"uow\_id - Unit of work ID monitor element" on page 1401  
"wlm\_queue\_time\_total - Workload manager total queue time monitor element" on page 1422  
"wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element" on page 1421  
"fcm\_tq\_rcv\_wait\_time - FCM table queue received wait time monitor element" on page 840  
"fcm\_message\_rcv\_wait\_time - FCM message received wait time monitor element" on page 824  
"fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element" on page 844  
"fcm\_message\_send\_wait\_time - FCM message send wait time monitor element" on page 828

"agent\_wait\_time - Agent wait time monitor element" on page 654  
"agent\_waits\_total - Total agent waits monitor element" on page 655  
"lock\_wait\_time - Time waited on locks monitor element" on page 924  
"lock\_waits - Lock waits monitor element" on page 928  
"direct\_read\_time - Direct read time monitor element" on page 792  
"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_write\_time - Direct write time monitor element" on page 798  
"direct\_write\_reqs - Direct write requests monitor element" on page 796  
"log\_buffer\_wait\_time - Log buffer wait time monitor element" on page 933  
"num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element" on page 975  
"log\_disk\_wait\_time - Log disk wait time monitor element" on page 935  
"log\_disk\_waits\_total - Total log disk waits monitor element" on page 936  
"tcpip\_rcv\_wait\_time - TCP/IP received wait time monitor element" on page 1298  
"tcpip\_rcvs\_total - TCP/IP receives total monitor element" on page 1299  
"client\_idle\_wait\_time - Client idle wait time monitor element" on page 712  
"ipc\_rcv\_wait\_time - Interprocess communication received wait time monitor element" on page 888  
"ipc\_rcvs\_total - Interprocess communication receives total monitor element" on page 889  
"ipc\_send\_wait\_time - Interprocess communication send wait time monitor element" on page 891  
"ipc\_sends\_total - Interprocess communication send total monitor element" on page 892  
"tcpip\_send\_wait\_time - TCP/IP send wait time monitor element" on page 1301  
"tcpip\_sends\_total - TCP/IP sends total monitor element" on page 1302  
"pool\_write\_time - Total buffer pool physical write time monitor element" on page 1128  
"pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113  
"audit\_file\_write\_wait\_time - Audit file write wait time monitor element" on page 680  
"audit\_file\_writes\_total - Total audit files written monitor element" on page 682  
"audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element" on page 683  
"audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element" on page 685  
"diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element" on page 788  
"diaglog\_writes\_total - Total diagnostic log file writes monitor element" on page 789  
"fcm\_send\_wait\_time - FCM send wait time monitor element" on page 836  
"fcm\_rcv\_wait\_time - FCM received wait time monitor element" on page 832  
"total\_wait\_time - Total wait time monitor element" on page 1386  
"rqsts\_completed\_total - Total requests completed monitor element" on page 1197  
"total\_rqst\_time - Total request time monitor element" on page 1362

"app\_rqsts\_completed\_total - Total application requests completed monitor element" on page 665

"total\_app\_rqst\_time - Total application request time monitor element" on page 1316

"total\_section\_sort\_proc\_time - Total section sort processing time monitor element" on page 1368

"total\_section\_sorts - Total section sorts monitor element" on page 1371

"total\_section\_sort\_time - Total section sort time monitor element" on page 1370

"rows\_read - Rows read monitor element" on page 1191

"rows\_modified - Rows modified monitor element" on page 1190

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120

"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136

"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123

"total\_cpu\_time - Total CPU time monitor element" on page 1330

"act\_completed\_total - Total completed activities monitor element" on page 637

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139

"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_xda\_writes - Buffer pool XDA data writes monitor element" on page 1141

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_writes - Direct writes to database monitor element" on page 800

"rows\_returned - Rows returned monitor element" on page 1193

"deadlocks - Deadlocks detected monitor element" on page 783

"lock\_timeouts - Number of lock timeouts monitor element" on page 920

"lock\_escals - Number of lock escalations monitor element" on page 905

"fcm\_sends\_total - FCM sends total monitor element" on page 838

"fcm\_recvs\_total - FCM receives total monitor element" on page 834

"fcm\_send\_volume - FCM send volume monitor element" on page 835

"fcm\_rcv\_volume - FCM received volume monitor element" on page 831  
"fcm\_message\_sends\_total - Total FCM message sends monitor element" on page 830  
"fcm\_message\_rcvs\_total - Total FCM message receives monitor element" on page 826  
"fcm\_message\_send\_volume - FCM message send volume monitor element" on page 827  
"fcm\_message\_rcv\_volume - FCM message received volume monitor element" on page 823  
"fcm\_tq\_sends\_total - FCM table queue send total monitor element" on page 846  
"fcm\_tq\_rcvs\_total - FCM table queue receives total monitor element" on page 842  
"fcm\_tq\_send\_volume - FCM table queue send volume monitor element" on page 843  
"fcm\_tq\_rcv\_volume - FCM table queue received volume monitor element" on page 839  
"tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396  
"tcpip\_send\_volume - TCP/IP send volume monitor element" on page 1300  
"tcpip\_rcv\_volume - TCP/IP received volume monitor element" on page 1297  
"ipc\_send\_volume - Interprocess communication send volume monitor element" on page 890  
"ipc\_rcv\_volume - Interprocess communication received volume monitor element" on page 887  
"post\_threshold\_sorts - Post threshold sorts monitor element" on page 1150  
"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144  
"sort\_overflows - Sort overflows monitor element" on page 1224  
"audit\_events\_total - Total audit events monitor element" on page 679  
"act\_rejected\_total - Total rejected activities monitor element" on page 640  
"act\_aborted\_total - Total aborted activities monitor element" on page 636  
"total\_sorts - Total sorts monitor element" on page 1375  
"total\_routine\_time - Total routine time monitor element" on page 1357  
"total\_compile\_proc\_time - Total compile processing time monitor element" on page 1322  
"total\_compilations - Total compilations monitor element" on page 1321  
"total\_compile\_time - Total compile time monitor element" on page 1323  
"total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element" on page 1340  
"total\_implicit\_compilations - Total implicit complications monitor element" on page 1339  
"total\_implicit\_compile\_time - Total implicit compile time monitor element" on page 1341  
"total\_runstats\_proc\_time - Total runtime statistics processing time monitor element" on page 1364  
"total\_runstats - Total runtime statistics monitor element" on page 1363

"total\_runstats\_time - Total runtime statistics time monitor element" on page 1365

"total\_reorg\_proc\_time - Total reorganization processing time monitor element" on page 1350

"total\_reorgs - Total reorganizations monitor element" on page 1352

"total\_reorg\_time - Total reorganization time monitor element" on page 1351

"total\_load\_proc\_time - Total load processing time monitor element" on page 1342

"total\_loads - Total loads monitor element" on page 1344

"total\_load\_time - Total load time monitor element" on page 1343

"total\_section\_proc\_time - Total section processing time monitor element" on page 1367

"total\_app\_section\_executions - Total application section executions monitor element" on page 1317

"total\_section\_time - Total section time monitor element" on page 1373

"total\_commit\_proc\_time - Total commits processing time monitor element" on page 1319

"total\_app\_commits - Total application commits monitor elements" on page 1314

"total\_commit\_time - Total commit time monitor element" on page 1320

"total\_rollback\_proc\_time - Total rollback processing time monitor element" on page 1353

"total\_app\_rollbacks - Total application rollbacks monitor element" on page 1315

"total\_rollback\_time - Total rollback time monitor element" on page 1354

"total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element" on page 1359

"total\_routine\_user\_code\_time - Total routine user code time monitor element" on page 1360

"thresh\_violations - Number of threshold violations monitor element" on page 1304

"num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element" on page 979

"total\_routine\_invocations - Total routine invocations monitor elements" on page 1355

"int\_commits - Internal commits monitor element" on page 880

"int\_rollbacks - Internal rollbacks monitor element" on page 882

"cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698

"cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699

"pkg\_cache\_inserts - Package cache inserts monitor element" on page 1020

"pkg\_cache\_lookups - Package cache lookups monitor element" on page 1021

"act\_rqsts\_total - Total activity requests monitor elements" on page 642

"total\_act\_wait\_time - Total activity wait time monitor element" on page 1312

"total\_act\_time - Total activity time monitor element" on page 1311

"lock\_wait\_time\_global - Lock wait time global monitor element" on page 926

"lock\_waits\_global - Lock waits global monitor element" on page 930

"reclaim\_wait\_time - Reclaim wait time monitor element" on page 1170

“spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element” on page 1236

“lock\_timeouts\_global - Lock timeouts global monitor element” on page 922

“lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element” on page 910

“lock\_escals\_locklist - Number of locklist lock escalations monitor element” on page 909

“lock\_escals\_global - Number of global lock escalations monitor element” on page 907

“cf\_wait\_time - cluster caching facility wait time monitor element” on page 703

“cf\_waits - Number of cluster caching facility waits monitor element” on page 703

“pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element” on page 1044

“pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element” on page 1045

“pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element” on page 1047

“pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element” on page 1042

“pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element” on page 1075

“pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements” on page 1077

“pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element” on page 1078

“pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element” on page 1074

“pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element” on page 1132

“pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element” on page 1134

“pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element” on page 1138

“pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element” on page 1131

“evmon\_wait\_time - Event monitor wait time monitor element” on page 814

“evmon\_waits\_total - Event monitor total waits monitor element” on page 816

“total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element” on page 1335

“total\_extended\_latch\_waits - Total extended latch waits monitor element” on page 1336

“total\_stats\_fabrication\_proc\_time - Total statistics fabrication processing time monitor element” on page 1377

“total\_stats\_fabrications - Total statistics fabrications monitor elements” on page 1379

“total\_stats\_fabrication\_time - Total statistics fabrication time monitor element” on page 1378

“total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element” on page 1381

“total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element” on page 1382

“total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements” on page 1380

“total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element” on page 1332

“pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element” on page 1089

“pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element” on page 1093

“pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element” on page 1111

“pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element” on page 1087

“pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element” on page 1091

“pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element” on page 1109

“app\_act\_completed\_total - Total successful external coordinator activities monitor element” on page 662

“app\_act\_aborted\_total - Total failed external coordinator activities monitor element” on page 661

“app\_act\_rejected\_total - Total rejected external coordinator activities monitor element” on page 664

### **uow\_package\_list logical data group**

“partition\_key - Partitioning key monitor element” on page 1016

“appl\_id - Application ID monitor element” on page 666

“invocation\_id - Invocation ID monitor element” on page 886

“nesting\_level - Nesting level monitor element” on page 967

“package\_elapsed\_time - Package elapsed time monitor element” on page 1005

“package\_id - Package identifier monitor element” on page 1005

“routine\_id - Routine ID monitor element” on page 1187

“uow\_id - Unit of work ID monitor element” on page 1401

“member - Database member monitor element” on page 957

### **utillocation logical data group**

“event\_id - Event ID monitor element” on page 810

“event\_timestamp - Event timestamp monitor element” on page 811

“member - Database member monitor element” on page 957

“event\_type - Event Type monitor element” on page 812

“utility\_invocation\_id - Utility invocation ID” on page 1413

“utility\_type - Utility Type” on page 1417

“device\_type - Device type” on page 787

“location\_type - Location type” on page 901

“location - Location” on page 900

## **utilphase logical data group**

- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "member - Database member monitor element" on page 957*
- "event\_type - Event Type monitor element" on page 812*
- "utility\_invocation\_id - Utility invocation ID" on page 1413*
- "utility\_type - Utility Type" on page 1417*
- "utility\_phase\_type - Utility phase type" on page 1415*
- "phase\_start\_event\_id - Phase start event ID" on page 1018*
- "phase\_start\_event\_timestamp - Phase start event timestamp" on page 1018*
- "objtype - Object type monitor element" on page 996*
- "object\_schema - Object schema monitor element" on page 992*
- "object\_name - Object name monitor element" on page 991*
- "utility\_phase\_detail - Utility phase detail" on page 1415*

## **utilstart logical data group**

- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "member - Database member monitor element" on page 957*
- "event\_type - Event Type monitor element" on page 812*
- "utility\_invocation\_id - Utility invocation ID" on page 1413*
- "utility\_type - Utility Type" on page 1417*
- "utility\_operation\_type - Utility operation type" on page 1414*
- "utility\_invoker\_type - Utility Invoker Type" on page 1413*
- "utility\_priority - Utility Priority" on page 1415*
- "utility\_start\_type - Utility start type" on page 1416*
- "objtype - Object type monitor element" on page 996*
- "object\_schema - Object schema monitor element" on page 992*
- "object\_name - Object name monitor element" on page 991*
- "num\_tbsps - Number of table spaces monitor element" on page 981*
- "tbsp\_names - Table space names" on page 1296*
- "utility\_detail - Utility detail" on page 1412*

## **utilstop logical data group**

- "event\_id - Event ID monitor element" on page 810*
- "event\_timestamp - Event timestamp monitor element" on page 811*
- "member - Database member monitor element" on page 957*
- "event\_type - Event Type monitor element" on page 812*
- "utility\_invocation\_id - Utility invocation ID" on page 1413*
- "utility\_type - Utility Type" on page 1417*
- "utility\_stop\_type - Utility stop type" on page 1417*
- "start\_event\_id - Start event ID" on page 1240*
- "start\_event\_timestamp - Start event timestamp" on page 1240*
- "sqlca - SQL Communications Area (SQLCA)" on page 1229*

---

## Event type mappings to logical data groups

For file and pipe event monitors, event monitor output consists of an ordered series of logical data groupings. Regardless of the event monitor type, the output records always contain the same starting logical data groups. These frame the logical data groups whose presence varies depending on the event types recorded by the event monitor.

For file and pipe event monitors, event records may be generated for any connection and may therefore appear in mixed order in the stream. This means that you may get a transaction event for Connection 1, immediately followed by a connection event for Connection 2. However, records belonging to a single connection or a single event will appear in their logical order. For example, a statement record (end of statement) always precedes a transaction record (end of UOW), if any. Similarly, a deadlock event record always precedes the deadlocked connection event records for each connection involved in the deadlock. The **application id** or **application handle (agent\_id)** can be used to match records with a connection.

Connection header events are normally written for each connection to the database. For deadlocks with details event monitors, they are only written when the deadlock occurs. In this case, connection header events are only written for participants in the deadlock and not for all connections to the database.

The logical data groupings are ordered according to four different levels: Monitor, Prolog, Contents, and Epilog. Following are detailed descriptions for each level, including the corresponding event types and logical data groups.

### Monitor

Information at the Monitor level is generated for all event monitors. It consists of event monitor metadata.

*Table 144. Event Monitor Data Stream: Monitor Section*

| Event type    | Logical data group      | Available information  |
|---------------|-------------------------|--|
| Monitor Level | event_log_stream_header | Identifies the version level and byte order of the event monitor. Applications can use this header to determine whether they can handle the evmon output stream. |

### Prolog

The Prolog information is generated when the event monitor is activated.

*Table 145. Event Monitor Data Stream: Prolog Section*

| Event type          | Logical data group | Available information  |
|---------------------|--------------------|--|
| Log Header          | event_log_header   | Characteristics of the trace, for example server type and memory layout. |
| Database Header     | event_db_header    | Database name, path and activation time.                                 |
| Event Monitor Start | event_start        | Time when the monitor was started or restarted.                          |

Table 145. Event Monitor Data Stream: Prolog Section (continued)

| Event type        | Logical data group | Available information  |
|-------------------|--------------------|--|
| Connection Header | event_connheader   | One for each current connection, includes connection time and application name. Event connection headers are only generated for connection, statement, transaction, and deadlock event monitors. Deadlocks with details event monitors produce connection headers only when a deadlock occurs. |

## Contents

Information specific to the event monitor's specified event types is presented in the Contents section.

Table 146. Event Monitor Data Stream: Contents Section

| Event type   | Logical data group          | Available information   |
|--|-----------------------------|---|
| Statement Event                                    | event_stmt                  | Statement level data, including text for dynamic statements. Statement event monitors do not log fetches.   |
| Subsection Event                                   | event_subsection            | Subsection level data.  |
| Transaction Event <sup>1</sup>                     | event_xact                  | Transaction level data.   |
| Connection Event                                   | event_conn                  | Connection level data.  |
| Deadlock Event                                     | event_deadlock              | Deadlock level data.  |
| Deadlocked Connection Event                        | event_dlconn                | One for each connection involved in the deadlock, includes applications involved and locks in contention.   |
| Deadlocked Connection Event with Details           | event_detailed_dlconn, lock | One for each connection involved in the deadlock, includes applications involved, locks in contention, current statement information, and other locks held by the application contention. |
| Overflow   | event_overflow              | Number of records lost - generated when writer cannot keep up with a (non-blocked) event monitor.   |
| Deadlocks with details history <sup>2</sup>        | event_stmt_history          | List of statements executed in any unit of work that was involved in a deadlock.  |
| Deadlocks with details history values <sup>2</sup> | event_data_value            | Parameter markers for a statement in the event_stmt_history list.   |

Table 146. Event Monitor Data Stream: Contents Section (continued)

| Event type           | Logical data group        | Available information   |
|----------------------|---------------------------|---|
| Activities           | event_activity            | List of activities that completed executing on the system or were captured before completion.   |
|                      | event_activitystmt        | Information about the statement the activity was executing if the activity type was a statement.  |
|                      | event_activityvals        | The data values used as input variables for each activity that is an SQL statement. These data values do not include LOB data, long data, or structured type data.                |
| Statistics           | event_scstats             | Statistics computed from the activities that executed within each service class, work class, or workload in the system, as well as statistics computed from the threshold queues. |
|                      | event_wcstats             |   |
|                      | event_wlstats             |   |
|                      | event_qstats              |   |
|                      | event_histogrambin        |   |
| Threshold violations | event_thresholdviolations | Information identifying the threshold violated and the time of violation.   |

- <sup>1</sup> This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR UNIT OF WORK statement to monitor transaction events.
- <sup>2</sup> This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Epilog

The Epilog information is generated during database deactivation (last application finished disconnecting):

Table 147. Event Monitor Data Stream: Epilog Section

| Event type        | Logical data group | Available information        |
|-------------------|--------------------|------------------------------|
| Database Event    | event_db           | Database manager level data. |
| Buffer Pool Event | event_bufferpool   | Buffer pool level data.      |
| Table Space Event | event_tablespace   | Table space level data.      |
| Table Event       | event_table        | Table level data.            |

## Logical data groups affected by COLLECT ACTIVITY DATA settings

The following table shows what logical data groups are collected when different COLLECT ACTIVITY DATA options are specified all types of WLM objects, including Service Subclass, Workload, Work Class (via a Work Action), and Threshold.

Table 148. COLLECT ACTIVITY DATA settings

| Setting for COLLECT ACTIVITY DATA | Logical data groups collected   |
|-----------------------------------|---|
| NONE                              | none  |
| WITHOUT DETAILS                   | event_activity<br>event_activitymetrics   |
| WITH DETAILS                      | event_activity<br>event_activitymetrics<br>event_activitystmt                       |
| WITH DETAILS AND VALUES           | event_activity<br>event_activitymetrics<br>event_activitystmt<br>event_activityvals |

## Snapshot monitor interface mappings to logical data groups

The following table lists several ways of accessing snapshot monitor data. All snapshot monitor data is stored in monitor elements, which are categorized by logical data groups. Each individual API request type, CLP command, and SQL administrative view only captures monitor data from a subset of all the logical data groups.

Each individual API request type, CLP command, and SQL administrative view listed in this table returns monitor elements from the logical data groups listed in the right-most column.

**Note:**

1. There are a number of API request types and CLP commands for which there are no corresponding SQL administrative view. For other API request types and CLP commands, individual SQL administrative views capture subsets of the associated logical data groups.
2. Some monitor elements are returned only if the associated monitor switch is set ON. See the individual monitor elements to determine if a required element is under switch control.

Table 149. Snapshot Monitor Interface Mappings to Logical Data Groups

| db2GetSnapshot API request type | CLP command   | SQL administrative view | Logical data groups |
|---------------------------------|---|-------------------------|---------------------|
| SQLMA_APPLINFO_ALL              | list applications<br>[show detail]                                  | APPLICATIONS            | appl_info           |
| SQLMA_DBASE_APPLINFO            | list applications<br>for database<br><i>dbname</i> [show<br>detail] | APPLICATIONS            | appl_info           |
| SQLMA_DCS_APPLINFO_ALL          | list dcs<br>applications [show<br>detail]                           |                         | dcs_appl_info       |

Table 149. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

| db2GetSnapshot API request type | CLP command  | SQL administrative view | Logical data groups           |
|---------------------------------|--|-------------------------|-------------------------------|
| SQLMA_DB2                       | get snapshot for dbm                               | SNAPDBM                 | db2                           |
|                                 |  | SNAPFCM                 | fc                            |
|                                 |  | SNAPFCMPART             | fc_node                       |
|                                 |  | SNAPUTIL                | utility_info                  |
|                                 |  | SNAPUTIL_PROGRESS       | progress, progress_info       |
|                                 |  | SNAPDBM_MEMORY_POOL     | memory_pool                   |
|                                 | get dbm monitor switches                           | SNAPSWITCHES            | switch_list                   |
| SQLMA_DBASE                     | get snapshot for database on <i>dbname</i>         | SNAPDB                  | dbase                         |
|                                 |  | SNAPDETAILLOG           | detail_log                    |
|                                 |  | ADMIN_GET_STORAGE_PATHS | db_storage_group              |
|                                 |  |                         | rollforward                   |
|                                 |  |                         | db_sto_path_info              |
|                                 |  | SNAPTbsp                | tablespace                    |
|                                 |  | SNAPDB_MEMORY_POOL      | memory_pool                   |
| SQLMA_DBASE_ALL                 | get snapshot for all databases                     | SNAPDB                  | dbase                         |
|                                 |  | ADMIN_GET_STORAGE_PATHS | db_storage_group              |
|                                 |  |                         | rollforward                   |
|                                 |  |                         | db_sto_path_info              |
|                                 |  | SNAPTbsp                | tablespace                    |
|                                 | SNAPDB_MEMORY_POOL                                 | memory_pool             |                               |
|                                 | list active databases                              | dbase                   |                               |
| SQLMA_DCS_DBASE                 | get snapshot for dcs database on <i>dbname</i>     |                         | dcs_dbase, stmt_transmissions |
| SQLMA_DCS_DBASE_ALL             | get snapshot for all dcs databases                 |                         | dcs_dbase, stmt_transmissions |
| SQLMA_DBASE_REMOTE              | get snapshot for remote database on <i>dbname</i>  |                         | dbase_remote                  |
| SQLMA_DBASE_REMOTE_ALL          | get snapshot for all remote databases              |                         | dbase_remote                  |
| SQLMA_APPL                      | get snapshot for application applid <i>appl-id</i> | SNAPAPPL                | appl                          |
|                                 |  | SNAPAGENT               | agent                         |
|                                 |  | SNAPAPPL_INFO           | appl_info                     |
|                                 |  | SNAPLOCKWAIT            | lock_wait                     |
|                                 |  | SNAPSTMT                | stmt                          |
|                                 |  | SNAPSUBSECTION          | subsection                    |
|                                 | SNAPAGENT_MEMORY_POOL                              | memory_pool             |                               |

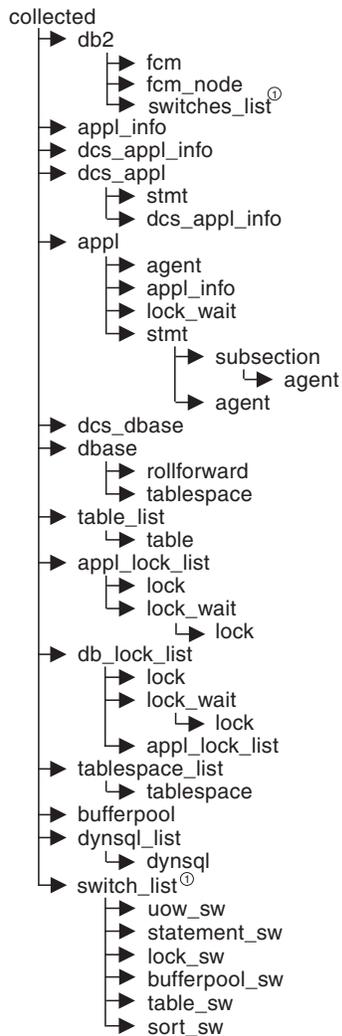
Table 149. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

| db2GetSnapshot API request type | CLP command   | SQL administrative view | Logical data groups |
|---------------------------------|---|-------------------------|---------------------|
| SQLMA_AGENT_ID                  | get snapshot for application<br>agentid<br><i>appl-handle</i>     | SNAPAGENT               | appl                |
|                                 |   | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTMT                | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
|                                 |   | SNAPAGENT_MEMORY_POOL   | memory pool         |
| SQLMA_DBASE_APPLS               | get snapshot for applications on<br><i>dbname</i>                 | SNAPAPPL                | appl                |
|                                 |   | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTMT                | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
|                                 |   | SNAPAGENT_MEMORY_POOL   | memory_pool         |
| SQLMA_APPL_ALL                  | get snapshot for all applications                                 | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPAGENT               | agent               |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_DCS_APPL                  | get snapshot for dcs application<br>applid <i>appl-id</i>         | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_DCS_APPL_ALL              | get snapshot for all dcs applications                             | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_DCS_APPL_HANDLE           | get snapshot for dcs application<br>agentid<br><i>appl-handle</i> | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_DCS_DBASE_APPLS           | get snapshot for dcs applications on<br><i>dbname</i>             | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_DBASE_APPLS_REMOTE        | get snapshot for remote applications on<br><i>dbname</i>          | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |
| SQLMA_APPL_REMOTE_ALL           | get snapshot for all remote applications                          | SNAPAGENT               | agent               |
|                                 |   | SNAPAPPL                | appl                |
|                                 |   | SNAPAPPL_INFO           | appl_info           |
|                                 |   | SNAPLOCKWAIT            | lock_wait           |
|                                 |   | SNAPSTATEMENT           | stmt                |
|                                 |   | SNAPSUBSECTION          | subsection          |

Table 149. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

| db2GetSnapshot API request type | CLP command   | SQL administrative view              | Logical data groups                       |
|---------------------------------|---|--------------------------------------|---|
| SQLMA_DBASE_TABLES              | get snapshot for tables on <i>dbname</i>                          | SNAPTAB                              | table                                     |
|                                 |   | SNAPTAB_REORG                        | table_reorg                               |
|                                 |   |                                      | table_list                                |
| SQLMA_APPL_LOCKS                | get snapshot for locks for application applid <i>appl-id</i>      | SNAPLOCK, SNAPAPPL, SNAPLOCKWAIT     | appl_lock_list, lock_wait, lock           |
| SQLMA_APPL_LOCKS_AGENT_ID       | get snapshot for locks for application agentid <i>appl-handle</i> | SNAPLOCK, SNAPAPPL, SNAPLOCKWAIT     | appl_lock_list, lock_wait, lock           |
| SQLMA_DBASE_LOCKS               | get snapshot for locks on <i>dbname</i>                           | SNAPLOCK                             | appl_lock_list, lock                      |
|                                 |   | SNAPLOCK, SNAPLOCKWAIT               | db_lock_list, lock_wait                   |
| SQLMA_DBASE_TABLESPACES         | get snapshot for tablespaces on <i>dbname</i>                     | SNAPTbsp                             | tablespace                                |
|                                 |   | SNAPTbspPART                         | tablespace, tablespace_nodeinfo           |
|                                 |   | SNAPTbsp QUIESCER                    | tablespace_quiescer, tablespace_nodeinfo  |
|                                 |   | SNAPCONTAINER                        | tablespace_container, tablespace_nodeinfo |
|                                 |   | SNAPTbsp_RANGE                       | tablespace_ranges, tablespace_nodeinfo    |
|                                 |   | tablespace_list, tablespace_nodeinfo |   |
| SQLMA_BUFFERPOOLS_ALL           | get snapshot for all bufferpools                                  | SNAPBP                               | bufferpool                                |
| SQLMA_DBASE_BUFFERPOOLS         | get snapshot for bufferpools on <i>dbname</i>                     | SNAPBP                               | bufferpool                                |
| SQLMA_DYNAMIC_SQL               | get snapshot for dynamic sql on <i>dbname</i>                     | SNAPDYN_SQL                          | dynsql                                    |
|                                 |   |                                      | dynsql_list                               |

The following figure shows the order that logical data groupings may appear in a snapshot data stream.



① Similar structures (lower level\_sw items are returned by db2, but are not shown in the figure)

Figure 18. Data Stream Hierarchy

**Note:** Times may be returned as part of any logical data grouping.

## Snapshot monitor logical data groups and monitor elements

The following sections list the logical data groupings and monitor elements that can be returned by snapshot monitoring.

- “agent logical data group” on page 604
- “appl logical data group” on page 604
- “appl\_id\_info logical data group” on page 607
- “appl\_info logical data group” on page 608
- “appl\_lock\_list logical data group” on page 608
- “appl\_remote logical data group” on page 609
- “bufferpool logical data group” on page 609
- “bufferpool\_nodeinfo logical data group” on page 611
- “collected logical data group” on page 611
- “db2 logical data group” on page 611

- “db\_lock\_list logical data group” on page 612
- “dbase logical data group” on page 612
- “dbase\_remote logical data group” on page 617
- “db\_storage\_group logical data group” on page 617
- “dcs\_appl logical data group” on page 618
- “dcs\_appl\_info logical data group” on page 619
- “dcs\_dbase logical data group” on page 620
- “dcs\_stmt logical data group” on page 622
- “detail\_log logical data group” on page 622
- “dynsql logical data group” on page 623
- “dynsql\_list logical data group” on page 624
- “fcm logical data group” on page 624
- “fcm\_node logical data group” on page 624
- “hadr logical data group” on page 624
- “lock logical data group” on page 625
- “lock\_wait logical data group” on page 625
- “memory\_pool logical data group” on page 626
- “progress logical data group” on page 626
- “progress\_list logical data group” on page 626
- “rollforward logical data group” on page 626
- “stmt logical data group” on page 626
- “stmt\_transmissions logical data group” on page 627
- “subsection logical data group” on page 629
- “table logical data group” on page 629
- “table\_list logical data group” on page 630
- “table\_reorg logical data group” on page 630
- “tablespace logical data group” on page 630
- “tablespace\_container logical data group” on page 632
- “tablespace\_list logical data group” on page 632
- “tablespace\_nodeinfo logical data group” on page 633
- “tablespace\_quiescer logical data group” on page 634
- “tablespace\_range logical data group” on page 634
- “utility\_info logical data group” on page 634

### **agent logical data group**

“agent\_pid - Engine dispatchable unit (EDU) identifier monitor element” on page 652

“lock\_timeout\_val - Lock timeout value monitor element” on page 920

### **appl logical data group**

“acc\_curs\_blk - Accepted Block Cursor Requests” on page 636

“agent\_sys\_cpu\_time - System CPU Time used by Agent” on page 653

“agent\_usr\_cpu\_time - User CPU Time used by Agent” on page 653

“agents\_stolen - Stolen Agents” on page 658

“appl\_con\_time - Connection Request Start Timestamp” on page 666

“appl\_idle\_time - Application Idle Time” on page 670

"appl\_priority - Application Agent Priority" on page 671  
 "appl\_priority\_type - Application Priority Type" on page 672  
 "associated\_agents\_top - Maximum Number of Associated Agents" on page 678  
 "authority\_bitmap - User authorization level monitor element" on page 687  
 "authority\_lvl - User authorization level monitor element" on page 688  
 "binds\_precompiles - Binds/Precompiles Attempted" on page 690  
 "cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698  
 "cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699  
 "cat\_cache\_overflows - Catalog Cache Overflows" on page 700  
 "commit\_sql\_stmts - Commit Statements Attempted" on page 721  
 "conn\_complete\_time - Connection Request Completion Timestamp" on page 735  
 "ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements" on page 781  
 "deadlocks - Deadlocks detected monitor element" on page 783  
 "direct\_read\_reqs - Direct read requests monitor element" on page 790  
 "direct\_read\_time - Direct read time monitor element" on page 792  
 "direct\_reads - Direct reads from database monitor element" on page 794  
 "direct\_write\_reqs - Direct write requests monitor element" on page 796  
 "direct\_write\_time - Direct write time monitor element" on page 798  
 "direct\_writes - Direct writes to database monitor element" on page 800  
 "dynamic\_sql\_stmts - Dynamic SQL Statements Attempted" on page 804  
 "failed\_sql\_stmts - Failed Statement Operations" on page 821  
 "hash\_join\_overflows - Hash Join Overflows" on page 867  
 "hash\_join\_small\_overflows - Hash Join Small Overflows" on page 867  
 "inbound\_comm\_address - Inbound Communication Address" on page 874  
 "int\_auto\_rebinds - Internal Automatic Rebinds" on page 879  
 "int\_commits - Internal commits monitor element" on page 880  
 "int\_deadlock\_rollback - Internal Rollbacks Due To Deadlock" on page 881  
 "int\_rollback - Internal rollbacks monitor element" on page 882  
 "int\_rows\_deleted - Internal Rows Deleted" on page 884  
 "int\_rows\_inserted - Internal Rows Inserted" on page 884  
 "int\_rows\_updated - Internal Rows Updated" on page 885  
 "last\_reset - Last Reset Timestamp" on page 896  
 "lock\_escalation - Lock escalation monitor element" on page 904  
 "lock\_timeout\_val - Lock timeout value monitor element" on page 920  
 "lock\_timeouts - Number of lock timeouts monitor element" on page 920  
 "lock\_wait\_time - Time waited on locks monitor element" on page 924  
 "lock\_waits - Lock waits monitor element" on page 928  
 "locks\_held - Locks held monitor element" on page 932  
 "locks\_waiting - Current agents waiting on locks monitor element" on page 933  
 "num\_agents - Number of Agents Working on a Statement" on page 971  
 "olap\_func\_overflows - OLAP Function Overflows monitor element" on page 997  
 "open\_loc\_curs - Open Local Cursors" on page 998  
 "open\_loc\_curs\_blk - Open Local Cursors with Blocking" on page 998

“open\_rem\_curs - Open Remote Cursors” on page 999

“open\_rem\_curs\_blk - Open Remote Cursors with Blocking” on page 999

“pkg\_cache\_inserts - Package cache inserts monitor element” on page 1020

“pkg\_cache\_lookups - Package cache lookups monitor element” on page 1021

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“prefetch\_wait\_time - Time waited for prefetch monitor element” on page 1151

“prev\_uow\_stop\_time - Previous Unit of Work Completion Timestamp” on page 1155

“priv\_workspace\_num\_overflows - Private Workspace Overflows” on page 1156

“priv\_workspace\_section\_inserts - Private Workspace Section Inserts” on page 1157

“priv\_workspace\_section\_lookups - Private Workspace Section Lookups” on page 1158

“priv\_workspace\_size\_top - Maximum Private Workspace Size” on page 1158

“rej\_curs\_blk - Rejected Block Cursor Requests” on page 1173

“rollback\_sql\_stmts - Rollback Statements Attempted” on page 1184

“rows\_deleted - Rows deleted monitor element” on page 1188

“rows\_inserted - Rows inserted monitor element” on page 1189

*"rows\_read - Rows read monitor element" on page 1191*  
*"rows\_selected - Rows Selected" on page 1195*  
*"rows\_updated - Rows updated monitor element" on page 1196*  
*"rows\_written - Rows Written" on page 1196*  
*"select\_sql\_stmts - Select SQL Statements Executed" on page 1203*  
*"shr\_workspace\_num\_overflows - Shared Workspace Overflows" on page 1211*  
*"shr\_workspace\_section\_inserts - Shared Workspace Section Inserts" on page 1212*  
*"shr\_workspace\_section\_lookups - Shared Workspace Section Lookups" on page 1213*  
*"shr\_workspace\_size\_top - Maximum Shared Workspace Size" on page 1213*  
*"sort\_overflows - Sort overflows monitor element" on page 1224*  
*"sql\_reqs\_since\_commit - SQL Requests Since Last Commit" on page 1228*  
*"static\_sql\_stmts - Static SQL Statements Attempted" on page 1241*  
*"total\_hash\_joins - Total Hash Joins" on page 1338*  
*"total\_hash\_loops - Total Hash Loops" on page 1338*  
*"total\_olap\_funcs - Total OLAP Functions monitor element" on page 1346*  
*"total\_sort\_time - Total sort time monitor element" on page 1374*  
*"total\_sorts - Total sorts monitor element" on page 1375*  
*"uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed" on page 1398*  
*"unread\_prefetch\_pages - Unread prefetch pages monitor element" on page 1399*  
*"uow\_comp\_status - Unit of Work Completion Status" on page 1399*  
*"uow\_elapsed\_time - Most Recent Unit of Work Elapsed Time" on page 1400*  
*"uow\_lock\_wait\_time - Total time unit of work waited on locks monitor element" on page 1403*  
*"uow\_log\_space\_used - Unit of work log space used monitor element" on page 1403*  
*"uow\_start\_time - Unit of work start timestamp monitor element" on page 1404*  
*"uow\_stop\_time - Unit of work stop timestamp monitor element" on page 1405*  
*"x\_lock\_escals - Exclusive lock escalations monitor element" on page 1429*  
*"xquery\_stmts - XQuery Statements Attempted" on page 1431*

### **appl\_id\_info logical data group**

*"agent\_id - Application handle (agent ID) monitor element" on page 650*  
*"appl\_id - Application ID monitor element" on page 666*  
*"appl\_name - Application name monitor element" on page 670*  
*"appl\_status - Application status monitor element" on page 673*  
*"auth\_id - Authorization ID" on page 686*  
*"client\_db\_alias - Database Alias Used by Application" on page 711*  
*"client\_prdid - Client product and version ID monitor element" on page 715*  
*"codepage\_id - ID of Code Page Used by Application" on page 718*  
*"db\_name - Database name monitor element" on page 773*  
*"db\_path - Database Path" on page 774*  
*"input\_db\_alias - Input Database Alias" on page 877*  
*"sequence\_no - Sequence number monitor element" on page 1204*

"status\_change\_time - Application Status Change Time" on page 1244

### **appl\_info logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_name - Application name monitor element" on page 670  
"appl\_section\_inserts - Section Inserts monitor element" on page 672  
"appl\_section\_lookups - Section Lookups" on page 673  
"appl\_status - Application status monitor element" on page 673  
"auth\_id - Authorization ID" on page 686  
"authority\_bitmap - User authorization level monitor element" on page 687  
"authority\_lvl - User authorization level monitor element" on page 688  
"client\_db\_alias - Database Alias Used by Application" on page 711  
"client\_pid - Client process ID monitor element" on page 713  
"client\_platform - Client operating platform monitor element" on page 714  
"client\_prdid - Client product and version ID monitor element" on page 715  
"client\_protocol - Client communication protocol monitor element" on page 716  
"codepage\_id - ID of Code Page Used by Application" on page 718  
"coord\_agent\_pid - Coordinator agent identifier monitor element" on page 748  
"coord\_node - Coordinating Node" on page 750  
"corr\_token - DRDA Correlation Token" on page 751  
"db\_name - Database name monitor element" on page 773  
"db\_path - Database Path" on page 774  
"execution\_id - User Login ID" on page 820  
"input\_db\_alias - Input Database Alias" on page 877  
"is\_system\_appl - Is System Application monitor element" on page 893  
"num\_assoc\_agents - Number of Associated Agents" on page 971  
"sequence\_no - Sequence number monitor element" on page 1204  
"session\_auth\_id - Session authorization ID monitor element" on page 1211  
"status\_change\_time - Application Status Change Time" on page 1244  
"territory\_code - Database Territory Code" on page 1303  
"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387  
"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388  
"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388  
"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389  
"workload\_id - Workload ID monitor element" on page 1426

### **appl\_lock\_list logical data group**

"agent\_id - Application handle (agent ID) monitor element" on page 650  
"appl\_id - Application ID monitor element" on page 666  
"appl\_name - Application name monitor element" on page 670  
"appl\_status - Application status monitor element" on page 673

*"auth\_id - Authorization ID" on page 686*  
*"client\_db\_alias - Database Alias Used by Application" on page 711*  
*"codepage\_id - ID of Code Page Used by Application" on page 718*  
*"lock\_wait\_time - Time waited on locks monitor element" on page 924*  
*"locks\_held - Locks held monitor element" on page 932*  
*"locks\_waiting - Current agents waiting on locks monitor element" on page 933*  
*"sequence\_no - Sequence number monitor element" on page 1204*  
*"session\_auth\_id - Session authorization ID monitor element" on page 1211*  
*"status\_change\_time - Application Status Change Time" on page 1244*

### **appl\_remote logical data group**

*"commit\_sql\_stmts - Commit Statements Attempted" on page 721*  
*"create\_nickname - Create Nicknames" on page 762*  
*"create\_nickname\_time - Create Nickname Response Time" on page 763*  
*"datasource\_name - Data Source Name" on page 768*  
*"db\_name - Database name monitor element" on page 773*  
*"delete\_sql\_stmts - Deletes" on page 786*  
*"delete\_time - Delete Response Time" on page 786*  
*"failed\_sql\_stmts - Failed Statement Operations" on page 821*  
*"insert\_sql\_stmts - Inserts" on page 877*  
*"insert\_time - Insert Response Time" on page 878*  
*"passthru\_time - Pass-Through Time" on page 1017*  
*"passthru - Pass-Through" on page 1017*  
*"remote\_lock\_time - Remote Lock Time" on page 1175*  
*"remote\_locks - Remote Locks" on page 1175*  
*"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184*  
*"rows\_deleted - Rows deleted monitor element" on page 1188*  
*"rows\_inserted - Rows inserted monitor element" on page 1189*  
*"rows\_selected - Rows Selected" on page 1195*  
*"rows\_updated - Rows updated monitor element" on page 1196*  
*"select\_sql\_stmts - Select SQL Statements Executed" on page 1203*  
*"select\_time - Query Response Time" on page 1203*  
*"sp\_rows\_selected - Rows Returned by Stored Procedures" on page 1227*  
*"stored\_proc\_time - Stored Procedure Time" on page 1263*  
*"stored\_procs - Stored Procedures" on page 1264*  
*"update\_sql\_stmts - Updates" on page 1407*  
*"update\_time - Update Response Time" on page 1407*

### **bufferpool logical data group**

*"block\_ios - Number of block I/O requests monitor element" on page 690*  
*"bp\_id - Buffer pool identifier monitor element" on page 693*  
*"bp\_name - Buffer pool name monitor element" on page 693*  
*"db\_name - Database name monitor element" on page 773*  
*"db\_path - Database Path" on page 774*  
*"direct\_read\_reqs - Direct read requests monitor element" on page 790*  
*"direct\_read\_time - Direct read time monitor element" on page 792*

"direct\_reads - Direct reads from database monitor element" on page 794

"direct\_write\_reqs - Direct write requests monitor element" on page 796

"direct\_write\_time - Direct write time monitor element" on page 798

"direct\_writes - Direct writes to database monitor element" on page 800

"files\_closed - Database files closed monitor element" on page 848

"input\_db\_alias - Input Database Alias" on page 877

"pages\_from\_block\_ios - Total number of pages read by block I/O monitor element" on page 1011

"pages\_from\_vectored\_ios - Total number of pages read by vectored I/O monitor element" on page 1012

"pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element" on page 1026

"pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element" on page 1027

"pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element" on page 1028

"pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element" on page 1031

"pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element" on page 1032

"pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element" on page 1033

"pool\_async\_read\_time - Buffer Pool Asynchronous Read Time" on page 1033

"pool\_async\_write\_time - Buffer pool asynchronous write time monitor element" on page 1034

"pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element" on page 1038

"pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element" on page 1038

"pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element" on page 1039

"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048

"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050

"pool\_data\_writes - Buffer pool data writes monitor element" on page 1052

"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080

"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082

"pool\_index\_writes - Buffer pool index writes monitor element" on page 1084

"pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element" on page 1087

"pool\_read\_time - Total buffer pool physical read time monitor element" on page 1113

"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116

"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“vectored\_ios - Number of vectored I/O requests monitor element” on page 1419

### **bufferpool\_nodeinfo logical data group**

“bp\_cur\_buffsz - Current Size of Buffer Pool” on page 693

“bp\_new\_buffsz - New Buffer Pool Size” on page 694

“bp\_pages\_left\_to\_remove - Number of Pages Left to Remove” on page 694

“bp\_tbsp\_use\_count - Number of Table Spaces Mapped to Buffer Pool” on page 694

“node\_number - Node Number” on page 970

### **collected logical data group**

“node\_number - Node Number” on page 970

“server\_db2\_type - Database Manager Type at Monitored (Server) Node” on page 1205

“server\_instance\_name - Server Instance Name” on page 1205

“server\_prdid - Server Product/Version ID” on page 1206

“server\_version - Server Version” on page 1207

“time\_stamp - Snapshot Time” on page 1309

“time\_zone\_disp - Time Zone Displacement” on page 1310

### **db2 logical data group**

“agents\_created\_empty\_pool - Agents Created Due to Empty Agent Pool” on page 656

“agents\_from\_pool - Agents Assigned From Pool” on page 657

“agents\_registered - Agents Registered” on page 657

“agents\_registered\_top - Maximum Number of Agents Registered” on page 658

“agents\_stolen - Stolen Agents” on page 658

“agents\_waiting\_on\_token - Agents Waiting for a Token” on page 659

“agents\_waiting\_top - Maximum Number of Agents Waiting monitor element” on page 659

“comm\_private\_mem - Committed Private Memory” on page 721

“con\_local\_databases - Local Databases with Current Connects” on page 724

*"coord\_agents\_top - Maximum Number of Coordinating Agents"* on page 749  
*"db2start\_time - Start Database Manager Timestamp"* on page 772  
*"db\_status - Status of database monitor element"* on page 775  
*"gw\_cons\_wait\_client - Number of Connections Waiting for the Client to Send Request"* on page 853  
*"gw\_cons\_wait\_host - Number of Connections Waiting for the Host to Reply"* on page 853  
*"gw\_cur\_cons - Current Number of Connections for DB2 Connect"* on page 854  
*"gw\_total\_cons - Total Number of Attempted Connections for DB2 Connect"* on page 855  
*"idle\_agents - Number of Idle Agents"* on page 872  
*"last\_reset - Last Reset Timestamp"* on page 896  
*"local\_cons - Local Connections"* on page 899  
*"local\_cons\_in\_exec - Local Connections Executing in the Database Manager"* on page 899  
*"max\_agent\_overflows - Maximum Agent Overflows"* on page 942  
*"num\_gw\_conn\_switches - Connection Switches"* on page 974  
*"num\_nodes\_in\_db2\_instance - Number of Nodes in Partition"* on page 980  
*"piped\_sorts\_accepted - Piped Sorts Accepted"* on page 1019  
*"piped\_sorts\_requested - Piped Sorts Requested"* on page 1019  
*"post\_threshold\_hash\_joins - Hash Join Threshold"* on page 1145  
*"post\_threshold\_olap\_funcs - OLAP Function Threshold monitor element"* on page 1146  
*"post\_threshold\_sorts - Post threshold sorts monitor element"* on page 1150  
*"product\_name - Product Name"* on page 1159  
*"rem\_cons\_in - Remote Connections To Database Manager"* on page 1174  
*"rem\_cons\_in\_exec - Remote Connections Executing in the Database Manager"* on page 1174  
*"service\_level - Service Level"* on page 1208  
*"smallest\_log\_avail\_node - Node with Least Available Log Space"* on page 1223  
*"sort\_heap\_allocated - Total Sort Heap Allocated"* on page 1223  
*"sort\_heap\_top - Sort private heap high watermark"* on page 1224

### **db\_lock\_list logical data group**

*"appls\_cur\_cons - Applications Connected Currently"* on page 677  
*"db\_name - Database name monitor element"* on page 773  
*"db\_path - Database Path"* on page 774  
*"input\_db\_alias - Input Database Alias"* on page 877  
*"locks\_held - Locks held monitor element"* on page 932  
*"locks\_waiting - Current agents waiting on locks monitor element"* on page 933

### **dbase logical data group**

*"active\_hash\_joins - Active hash joins"* on page 644  
*"active\_olap\_funcs - Active OLAP Functions monitor element"* on page 644  
*"active\_sorts - Active Sorts"* on page 644  
*"agents\_top - Number of Agents Created"* on page 658  
*"appl\_id\_oldest\_xact - Application with Oldest Transaction"* on page 669

*"appl\_section\_inserts - Section Inserts monitor element" on page 672*  
*"appl\_section\_lookups - Section Lookups" on page 673*  
*"appls\_cur\_cons - Applications Connected Currently" on page 677*  
*"appls\_in\_db2 - Applications Executing in the Database Currently" on page 677*  
*"async\_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 678*  
*"binds\_precompiles - Binds/Precompiles Attempted" on page 690*  
*"blocks\_pending\_cleanup - Pending cleanup rolled-out blocks monitor element" on page 692*  
*"cat\_cache\_inserts - Catalog cache inserts monitor element" on page 698*  
*"cat\_cache\_lookups - Catalog cache lookups monitor element" on page 699*  
*"cat\_cache\_overflows - Catalog Cache Overflows" on page 700*  
*"cat\_cache\_size\_top - Catalog cache high watermark monitor element" on page 701*  
*"catalog\_node - Catalog Node Number" on page 702*  
*"catalog\_node\_name - Catalog Node Network Name" on page 702*  
*"commit\_sql\_stmts - Commit Statements Attempted" on page 721*  
*"connections\_top - Maximum Number of Concurrent Connections" on page 737*  
*"coord\_agents\_top - Maximum Number of Coordinating Agents" on page 749*  
*"db\_conn\_time - Database activation timestamp monitor element" on page 772*  
*"db\_heap\_top - Maximum Database Heap Allocated" on page 773*  
*"db\_location - Database Location" on page 773*  
*"db\_name - Database name monitor element" on page 773*  
*"db\_path - Database Path" on page 774*  
*"db\_status - Status of database monitor element" on page 775*  
*"ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements" on page 781*  
*"deadlocks - Deadlocks detected monitor element" on page 783*  
*"direct\_read\_reqs - Direct read requests monitor element" on page 790*  
*"direct\_read\_time - Direct read time monitor element" on page 792*  
*"direct\_reads - Direct reads from database monitor element" on page 794*  
*"direct\_write\_reqs - Direct write requests monitor element" on page 796*  
*"direct\_write\_time - Direct write time monitor element" on page 798*  
*"direct\_writes - Direct writes to database monitor element" on page 800*  
*"dynamic\_sql\_stmts - Dynamic SQL Statements Attempted" on page 804*  
*"failed\_sql\_stmts - Failed Statement Operations" on page 821*  
*"files\_closed - Database files closed monitor element" on page 848*  
*"hash\_join\_overflows - Hash Join Overflows" on page 867*  
*"hash\_join\_small\_overflows - Hash Join Small Overflows" on page 867*  
*"input\_db\_alias - Input Database Alias" on page 877*  
*"int\_auto\_rebinds - Internal Automatic Rebinds" on page 879*  
*"int\_commits - Internal commits monitor element" on page 880*  
*"int\_deadlock\_rollbacks - Internal Rollbacks Due To Deadlock" on page 881*  
*"int\_rollbacks - Internal rollbacks monitor element" on page 882*  
*"int\_rows\_deleted - Internal Rows Deleted" on page 884*  
*"int\_rows\_inserted - Internal Rows Inserted" on page 884*

"int\_rows\_updated - Internal Rows Updated" on page 885  
 "last\_backup - Last Backup Timestamp" on page 894  
 "last\_reset - Last Reset Timestamp" on page 896  
 "lock\_escals - Number of lock escalations monitor element" on page 905  
 "lock\_list\_in\_use - Total lock list memory in use monitor element" on page 912  
 "lock\_timeouts - Number of lock timeouts monitor element" on page 920  
 "lock\_wait\_time - Time waited on locks monitor element" on page 924  
 "lock\_waits - Lock waits monitor element" on page 928  
 "locks\_held - Locks held monitor element" on page 932  
 "locks\_waiting - Current agents waiting on locks monitor element" on page 933  
 "log\_held\_by\_dirty\_pages - Amount of Log Space Accounted for by Dirty Pages" on page 937  
 "log\_read\_time - Log Read Time" on page 938  
 "log\_reads - Number of Log Pages Read" on page 938  
 "log\_to\_redo\_for\_recovery - Amount of Log to be Redone for Recovery" on page 939  
 "log\_write\_time - Log Write Time" on page 940  
 "log\_writes - Number of Log Pages Written" on page 940  
 "num\_assoc\_agents - Number of Associated Agents" on page 971  
 "num\_db\_storage\_paths - Number of automatic storage paths" on page 973  
 "num\_indoubt\_trans - Number of Indoubt Transactions" on page 975  
 "num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element" on page 975  
 "num\_log\_data\_found\_in\_buffer - Number of Log Data Found In Buffer" on page 977  
 "num\_log\_part\_page\_io - Number of Partial Log Page Writes" on page 977  
 "num\_log\_read\_io - Number of Log Reads" on page 978  
 "num\_log\_write\_io - Number of Log Writes" on page 978  
 "olap\_func\_overflows - OLAP Function Overflows monitor element" on page 997  
 "pkg\_cache\_inserts - Package cache inserts monitor element" on page 1020  
 "pkg\_cache\_lookups - Package cache lookups monitor element" on page 1021  
 "pkg\_cache\_num\_overflows - Package Cache Overflows" on page 1023  
 "pkg\_cache\_size\_top - Package cache high watermark" on page 1023  
 "pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element" on page 1026  
 "pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element" on page 1027  
 "pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element" on page 1028  
 "pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element" on page 1031  
 "pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element" on page 1032  
 "pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element" on page 1033  
 "pool\_async\_read\_time - Buffer Pool Asynchronous Read Time" on page 1033

“pool\_async\_write\_time - Buffer pool asynchronous write time monitor element” on page 1034

“pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element” on page 1038

“pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element” on page 1038

“pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element” on page 1039

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_drty\_pg\_steal\_clns - Buffer pool victim page cleaners triggered monitor element” on page 1055

“pool\_drty\_pg\_thrsh\_clns - Buffer pool threshold cleaners triggered monitor element” on page 1056

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_lsn\_gap\_clns - Buffer pool log space cleaners triggered monitor element” on page 1086

“pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element” on page 1087

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“post\_shrthreshold\_hash\_joins - Post threshold hash joins” on page 1143

"post\_shrthreshold\_sorts - Post shared threshold sorts monitor element" on page 1144

"priv\_workspace\_num\_overflows - Private Workspace Overflows" on page 1156

"priv\_workspace\_section\_inserts - Private Workspace Section Inserts" on page 1157

"priv\_workspace\_section\_lookups - Private Workspace Section Lookups" on page 1158

"priv\_workspace\_size\_top - Maximum Private Workspace Size" on page 1158

"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184

"rows\_deleted - Rows deleted monitor element" on page 1188

"rows\_inserted - Rows inserted monitor element" on page 1189

"rows\_read - Rows read monitor element" on page 1191

"rows\_selected - Rows Selected" on page 1195

"rows\_updated - Rows updated monitor element" on page 1196

"sec\_log\_used\_top - Maximum Secondary Log Space Used" on page 1199

"sec\_logs\_allocated - Secondary Logs Allocated Currently" on page 1200

"select\_sql\_stmts - Select SQL Statements Executed" on page 1203

"server\_platform - Server Operating System" on page 1206

"shr\_workspace\_num\_overflows - Shared Workspace Overflows" on page 1211

"shr\_workspace\_section\_inserts - Shared Workspace Section Inserts" on page 1212

"shr\_workspace\_section\_lookups - Shared Workspace Section Lookups" on page 1213

"shr\_workspace\_size\_top - Maximum Shared Workspace Size" on page 1213

"sort\_heap\_allocated - Total Sort Heap Allocated" on page 1223

"sort\_overflows - Sort overflows monitor element" on page 1224

"sort\_shrheap\_allocated - Sort Share Heap Currently Allocated" on page 1226

"sort\_shrheap\_top - Sort share heap high watermark" on page 1226

"static\_sql\_stmts - Static SQL Statements Attempted" on page 1241

"stats\_cache\_size - Size of statistics cache monitor element" on page 1242

"stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element" on page 1243

"stats\_fabrications - Total number of statistics fabrications monitor elements" on page 1244

"sync\_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 1265

"sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 1266

"tot\_log\_used\_top - Maximum Total Log Space Used" on page 1310

"total\_cons - Connects Since Database Activation" on page 1324

"total\_hash\_joins - Total Hash Joins" on page 1338

"total\_hash\_loops - Total Hash Loops" on page 1338

"total\_log\_available - Total Log Available" on page 1345

"total\_log\_used - Total Log Space Used" on page 1345

"total\_olap\_funcs - Total OLAP Functions monitor element" on page 1346

"total\_sec\_cons - Secondary Connections" on page 1366

"total\_sort\_time - Total sort time monitor element" on page 1374

“total\_sorts - Total sorts monitor element” on page 1375  
“uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed” on page 1398  
“unread\_prefetch\_pages - Unread prefetch pages monitor element” on page 1399  
“x\_lock\_escals - Exclusive lock escalations monitor element” on page 1429  
“xquery\_stmts - XQuery Statements Attempted” on page 1431

### **dbase\_remote logical data group**

“commit\_sql\_stmts - Commit Statements Attempted” on page 721  
“create\_nickname - Create Nicknames” on page 762  
“create\_nickname\_time - Create Nickname Response Time” on page 763  
“datasource\_name - Data Source Name” on page 768  
“db\_name - Database name monitor element” on page 773  
“delete\_sql\_stmts - Deletes” on page 786  
“delete\_time - Delete Response Time” on page 786  
“disconnects - Disconnects” on page 803  
“failed\_sql\_stmts - Failed Statement Operations” on page 821  
“insert\_sql\_stmts - Inserts” on page 877  
“insert\_time - Insert Response Time” on page 878  
“passthru\_time - Pass-Through Time” on page 1017  
“passthru - Pass-Through” on page 1017  
“remote\_lock\_time - Remote Lock Time” on page 1175  
“remote\_locks - Remote Locks” on page 1175  
“rollback\_sql\_stmts - Rollback Statements Attempted” on page 1184  
“rows\_deleted - Rows deleted monitor element” on page 1188  
“rows\_inserted - Rows inserted monitor element” on page 1189  
“rows\_selected - Rows Selected” on page 1195  
“rows\_updated - Rows updated monitor element” on page 1196  
“select\_sql\_stmts - Select SQL Statements Executed” on page 1203  
“select\_time - Query Response Time” on page 1203  
“sp\_rows\_selected - Rows Returned by Stored Procedures” on page 1227  
“stored\_proc\_time - Stored Procedure Time” on page 1263  
“stored\_procs - Stored Procedures” on page 1264  
“total\_cons - Connects Since Database Activation” on page 1324  
“update\_sql\_stmts - Updates” on page 1407  
“update\_time - Update Response Time” on page 1407

### **db\_storage\_group logical data group**

“fs\_id - Unique file system identification number monitor element” on page 850  
“fs\_total\_size - Total size of a file system monitor element” on page 850  
“fs\_used\_size - Amount of space used on a file system monitor element” on page 851  
“node\_number - Node Number” on page 970  
“sto\_path\_free\_sz - Automatic storage path free space monitor element” on page 1262

## **dc\_s\_appl logical data group**

- "appl\_idle\_time - Application Idle Time" on page 670*
- "commit\_sql\_stmts - Commit Statements Attempted" on page 721*
- "elapsed\_exec\_time - Statement Execution Elapsed Time" on page 806*
- "failed\_sql\_stmts - Failed Statement Operations" on page 821*
- "gw\_con\_time - DB2 Connect Gateway First Connect Initiated" on page 852*
- "gw\_exec\_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 854*
- "host\_response\_time - Host Response Time" on page 871*
- "inbound\_bytes\_received - Inbound Number of Bytes Received" on page 873*
- "inbound\_bytes\_sent - Inbound Number of Bytes Sent" on page 873*
- "last\_reset - Last Reset Timestamp" on page 896*
- "max\_data\_received\_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 945*
- "max\_data\_received\_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 946*
- "max\_data\_received\_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 946*
- "max\_data\_received\_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 947*
- "max\_data\_received\_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 947*
- "max\_data\_received\_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 948*
- "max\_data\_received\_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 948*
- "max\_data\_received\_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 948*
- "max\_data\_received\_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 949*
- "max\_data\_received\_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 949*
- "max\_data\_received\_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 950*
- "max\_data\_sent\_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 950*
- "max\_data\_sent\_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 950*
- "max\_data\_sent\_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 951*
- "max\_data\_sent\_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 951*
- "max\_data\_sent\_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 952*
- "max\_data\_sent\_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 952*
- "max\_data\_sent\_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 953*

*"max\_data\_sent\_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 953*

*"max\_data\_sent\_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 954*

*"max\_data\_sent\_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 954*

*"max\_data\_sent\_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 954*

*"max\_network\_time\_100\_ms - Number of Statements with Network Time between 16 and 100 ms" on page 955*

*"max\_network\_time\_16\_ms - Number of Statements with Network Time between 4 and 16 ms" on page 955*

*"max\_network\_time\_1\_ms - Number of Statements with Network Time of up to 1 ms" on page 956*

*"max\_network\_time\_4\_ms - Number of Statements with Network Time between 1 and 4 ms" on page 956*

*"max\_network\_time\_500\_ms - Number of Statements with Network Time between 100 and 500 ms" on page 957*

*"max\_network\_time\_gt500\_ms - Number of Statements with Network Time greater than 500 ms" on page 957*

*"network\_time\_bottom - Minimum Network Time for Statement" on page 968*

*"network\_time\_top - Maximum Network Time for Statement" on page 969*

*"open\_cursors - Number of Open Cursors" on page 998*

*"outbound\_bytes\_received - Outbound Number of Bytes Received" on page 1001*

*"outbound\_bytes\_sent - Outbound Number of Bytes Sent" on page 1002*

*"prev\_uow\_stop\_time - Previous Unit of Work Completion Timestamp" on page 1155*

*"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184*

*"rows\_selected - Rows Selected" on page 1195*

*"sql\_stmts - Number of SQL Statements Attempted" on page 1229*

*"tpmon\_acc\_str - TP monitor client accounting string monitor element" on page 1387*

*"tpmon\_client\_app - TP monitor client application name monitor element" on page 1388*

*"tpmon\_client\_userid - TP monitor client user ID monitor element" on page 1388*

*"tpmon\_client\_wkstn - TP monitor client workstation name monitor element" on page 1389*

*"uow\_comp\_status - Unit of Work Completion Status" on page 1399*

*"uow\_elapsed\_time - Most Recent Unit of Work Elapsed Time" on page 1400*

*"uow\_start\_time - Unit of work start timestamp monitor element" on page 1404*

*"uow\_stop\_time - Unit of work stop timestamp monitor element" on page 1405*

*"xid - Transaction ID" on page 1431*

### **dc\_s\_appl\_info logical data group**

*"agent\_id - Application handle (agent ID) monitor element" on page 650*

*"agent\_status - DCS Application Agents" on page 652*

*"appl\_id - Application ID monitor element" on page 666*

"appl\_name - Application name monitor element" on page 670  
 "auth\_id - Authorization ID" on page 686  
 "client\_pid - Client process ID monitor element" on page 713  
 "client\_platform - Client operating platform monitor element" on page 714  
 "client\_prdid - Client product and version ID monitor element" on page 715  
 "client\_protocol - Client communication protocol monitor element" on page 716  
 "codepage\_id - ID of Code Page Used by Application" on page 718  
 "dcs\_appl\_status - DCS application status monitor element" on page 780  
 "dcs\_db\_name - DCS Database Name" on page 780  
 "execution\_id - User Login ID" on page 820  
 "gw\_db\_alias - Database Alias at the Gateway" on page 854  
 "host\_ccsid - Host Coded Character Set ID" on page 869  
 "host\_db\_name - Host Database Name" on page 870  
 "host\_prdid - Host Product/Version ID" on page 871  
 "inbound\_comm\_address - Inbound Communication Address" on page 874  
 "outbound\_appl\_id - Outbound Application ID" on page 1001  
 "outbound\_comm\_address - Outbound Communication Address" on page 1003  
 "outbound\_comm\_protocol - Outbound Communication Protocol" on page 1004  
 "outbound\_sequence\_no - Outbound Sequence Number" on page 1004  
 "sequence\_no - Sequence number monitor element" on page 1204  
 "status\_change\_time - Application Status Change Time" on page 1244

### **dcs\_dbase logical data group**

"commit\_sql\_stmts - Commit Statements Attempted" on page 721  
 "con\_elapsed\_time - Most Recent Connection Elapsed Time" on page 724  
 "con\_response\_time - Most Recent Response Time for Connect" on page 725  
 "dcs\_db\_name - DCS Database Name" on page 780  
 "elapsed\_exec\_time - Statement Execution Elapsed Time" on page 806  
 "failed\_sql\_stmts - Failed Statement Operations" on page 821  
 "gw\_comm\_error\_time - Communication Error Time" on page 852  
 "gw\_comm\_errors - Communication Errors" on page 852  
 "gw\_con\_time - DB2 Connect Gateway First Connect Initiated" on page 852  
 "gw\_connections\_top - Maximum Number of Concurrent Connections to Host Database" on page 852  
 "gw\_cons\_wait\_client - Number of Connections Waiting for the Client to Send Request" on page 853  
 "gw\_cons\_wait\_host - Number of Connections Waiting for the Host to Reply" on page 853  
 "gw\_cur\_cons - Current Number of Connections for DB2 Connect" on page 854  
 "gw\_total\_cons - Total Number of Attempted Connections for DB2 Connect" on page 855  
 "host\_db\_name - Host Database Name" on page 870  
 "host\_response\_time - Host Response Time" on page 871  
 "inbound\_bytes\_received - Inbound Number of Bytes Received" on page 873  
 "last\_reset - Last Reset Timestamp" on page 896

*“max\_data\_received\_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes” on page 945*

*“max\_data\_received\_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes” on page 946*

*“max\_data\_received\_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes” on page 946*

*“max\_data\_received\_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes” on page 947*

*“max\_data\_received\_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes” on page 947*

*“max\_data\_received\_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element” on page 948*

*“max\_data\_received\_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes” on page 948*

*“max\_data\_received\_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes” on page 948*

*“max\_data\_received\_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element” on page 949*

*“max\_data\_received\_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes” on page 949*

*“max\_data\_received\_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes” on page 950*

*“max\_data\_sent\_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes” on page 950*

*“max\_data\_sent\_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes” on page 950*

*“max\_data\_sent\_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes” on page 951*

*“max\_data\_sent\_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes” on page 951*

*“max\_data\_sent\_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes” on page 952*

*“max\_data\_sent\_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes” on page 952*

*“max\_data\_sent\_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes” on page 953*

*“max\_data\_sent\_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes” on page 953*

*“max\_data\_sent\_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes” on page 954*

*“max\_data\_sent\_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes” on page 954*

*“max\_data\_sent\_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes” on page 954*

*“max\_network\_time\_100\_ms - Number of Statements with Network Time between 16 and 100 ms” on page 955*

*“max\_network\_time\_16\_ms - Number of Statements with Network Time between 4 and 16 ms” on page 955*

*“max\_network\_time\_1\_ms - Number of Statements with Network Time of up to 1 ms” on page 956*

"max\_network\_time\_4\_ms - Number of Statements with Network Time between 1 and 4 ms" on page 956  
"max\_network\_time\_500\_ms - Number of Statements with Network Time between 100 and 500 ms" on page 957  
"max\_network\_time\_gt500\_ms - Number of Statements with Network Time greater than 500 ms" on page 957  
"network\_time\_bottom - Minimum Network Time for Statement" on page 968  
"network\_time\_top - Maximum Network Time for Statement" on page 969  
"outbound\_bytes\_sent - Outbound Number of Bytes Sent" on page 1002  
"rollback\_sql\_stmts - Rollback Statements Attempted" on page 1184  
"rows\_selected - Rows Selected" on page 1195  
"sql\_stmts - Number of SQL Statements Attempted" on page 1229

### **dc\_s\_stmt logical data group**

"blocking\_cursor - Blocking Cursor" on page 691  
"creator - Application Creator" on page 763  
"elapsed\_exec\_time - Statement Execution Elapsed Time" on page 806  
"fetch\_count - Number of Successful Fetches" on page 847  
"gw\_exec\_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 854  
"host\_response\_time - Host Response Time" on page 871  
"inbound\_bytes\_received - Inbound Number of Bytes Received" on page 873  
"inbound\_bytes\_sent - Inbound Number of Bytes Sent" on page 873  
"num\_transmissions - Number of Transmissions" on page 982  
"num\_transmissions\_group - Number of Transmissions Group" on page 983  
"outbound\_bytes\_received - Outbound Number of Bytes Received" on page 1001  
"outbound\_bytes\_sent - Outbound Number of Bytes Sent" on page 1002  
"package\_name - Package name monitor element" on page 1006  
"query\_card\_estimate - Query Number of Rows Estimate" on page 1163  
"query\_cost\_estimate - Query cost estimate monitor element" on page 1163  
"section\_number - Section number monitor element" on page 1201  
"stmt\_elapsed\_time - Most Recent Statement Elapsed Time" on page 1245  
"stmt\_operation/operation - Statement operation monitor element" on page 1250  
"stmt\_start - Statement Operation Start Timestamp" on page 1254  
"stmt\_stop - Statement Operation Stop Timestamp" on page 1254  
"stmt\_text - SQL statement text monitor element" on page 1255

### **detail\_log logical data group**

"current\_active\_log - Current Active Log File Number" on page 765  
"current\_archive\_log - Current Archive Log File Number" on page 765  
"first\_active\_log - First Active Log File Number" on page 848  
"last\_active\_log - Last Active Log File Number" on page 893  
"node\_number - Node Number" on page 970

## **dynsql logical data group**

- “fetch\_count - Number of Successful Fetches” on page 847
- “insert\_timestamp - Insert timestamp monitor element” on page 878
- “int\_rows\_deleted - Internal Rows Deleted” on page 884
- “int\_rows\_inserted - Internal Rows Inserted” on page 884
- “int\_rows\_updated - Internal Rows Updated” on page 885
- “num\_compilations - Statement Compilations” on page 972
- “num\_executions - Statement executions monitor element” on page 973
- “pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048
- “pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050
- “pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080
- “pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082
- “pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116
- “pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118
- “pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120
- “pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121
- “pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123
- “pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125
- “pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136
- “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139
- “prep\_time\_best - Statement best preparation time monitor element” on page 1155
- “prep\_time\_worst - Statement worst preparation time monitor element” on page 1155
- “rows\_read - Rows read monitor element” on page 1191
- “rows\_written - Rows Written” on page 1196
- “sort\_overflows - Sort overflows monitor element” on page 1224
- “stats\_fabricate\_time - Total time spent on statistics fabrication activities monitor element” on page 1243
- “stmt\_pkgcache\_id - Statement package cache identifier monitor element” on page 1251
- “stmt\_sorts - Statement Sorts” on page 1253
- “stmt\_text - SQL statement text monitor element” on page 1255
- “sync\_runstats\_time - Total time spent on synchronous RUNSTATS activities monitor element” on page 1266

"total\_exec\_time - Elapsed statement execution time monitor element" on page 1334

"total\_sort\_time - Total sort time monitor element" on page 1374

"total\_sys\_cpu\_time - Total system CPU time for a statement monitor element" on page 1383

"total\_usr\_cpu\_time - Total user CPU time for a statement monitor element" on page 1385

### **dynsql\_list logical data group**

"db\_name - Database name monitor element" on page 773

"db\_path - Database Path" on page 774

### **fcv logical data group**

"buff\_free - FCM Buffers Currently Free" on page 695

"buff\_free\_bottom - Minimum FCM Buffers Free" on page 695

"buff\_max - Maximum possible number of FCM buffers monitor element" on page 696

"buff\_total - Number of currently allocated FCM buffers monitor element" on page 696

"ch\_free - Channels Currently Free" on page 707

"ch\_free\_bottom - Minimum Channels Free" on page 707

"ch\_max - Maximum possible number of FCM channels monitor element" on page 708

"ch\_total - Number of currently allocated FCM channels monitor element" on page 708

### **fcv\_node logical data group**

"connection\_status - Connection Status" on page 736

"node\_number - Node Number" on page 970

"total\_buffers\_rcvd - Total FCM Buffers Received" on page 1318

"total\_buffers\_sent - Total FCM Buffers Sent" on page 1318

### **hadr logical data group**

"hadr\_connect\_status - HADR Connection Status monitor element" on page 855

"hadr\_connect\_time - HADR Connection Time monitor element" on page 856

"hadr\_heartbeat - HADR Heartbeat monitor element" on page 857

"hadr\_local\_host - HADR Local Host monitor element" on page 857

"hadr\_local\_service - HADR Local Service monitor element" on page 858

"hadr\_log\_gap - HADR Log Gap" on page 858

"hadr\_primary\_log\_file - HADR Primary Log File monitor element" on page 860

"hadr\_primary\_log\_lsn - HADR Primary Log LSN monitor element" on page 860

"hadr\_primary\_log\_page - HADR Primary Log Page monitor element" on page 861

"hadr\_remote\_host - HADR Remote Host monitor element" on page 861

"hadr\_remote\_instance - HADR Remote Instance monitor element" on page 862

"hadr\_remote\_service - HADR Remote Service monitor element" on page 862

*"hadr\_role - HADR Role"* on page 863  
*"hadr\_standby\_log\_file - HADR Standby Log File monitor element"* on page 863  
*"hadr\_standby\_log\_lsn - HADR Standby Log LSN monitor element"* on page 864  
*"hadr\_standby\_log\_page - HADR Standby Log Page monitor element"* on page 864  
*"hadr\_state - HADR State monitor element"* on page 864  
*"hadr\_syncmode - HADR Synchronization Mode monitor element"* on page 865  
*"hadr\_timeout - HADR Timeout monitor element"* on page 866

## **lock logical data group**

*"data\_partition\_id - Data partition identifier monitor element"* on page 767  
*"lock\_attributes - Lock attributes monitor element"* on page 901  
*"lock\_count - Lock count monitor element"* on page 902  
*"lock\_current\_mode - Original lock mode before conversion monitor element"* on page 903  
*"lock\_escalation - Lock escalation monitor element"* on page 904  
*"lock\_hold\_count - Lock hold count monitor element"* on page 911  
*"lock\_mode - Lock mode monitor element"* on page 912  
*"lock\_name - Lock name monitor element"* on page 914  
*"lock\_object\_name - Lock Object Name"* on page 915  
*"lock\_object\_type - Lock object type waited on monitor element"* on page 916  
*"lock\_release\_flags - Lock release flags monitor element"* on page 918  
*"lock\_status - Lock status monitor element"* on page 919  
*"node\_number - Node Number"* on page 970  
*"table\_file\_id - Table file ID monitor element"* on page 1268  
*"table\_name - Table name monitor element"* on page 1269  
*"table\_schema - Table schema name monitor element"* on page 1271  
*"tablespace\_name - Table space name monitor element"* on page 1279

## **lock\_wait logical data group**

*"agent\_id\_holding\_lock - Agent ID Holding Lock"* on page 651  
*"appl\_id\_holding\_lk - Application ID Holding Lock"* on page 668  
*"data\_partition\_id - Data partition identifier monitor element"* on page 767  
*"lock\_attributes - Lock attributes monitor element"* on page 901  
*"lock\_current\_mode - Original lock mode before conversion monitor element"* on page 903  
*"lock\_escalation - Lock escalation monitor element"* on page 904  
*"lock\_mode - Lock mode monitor element"* on page 912  
*"lock\_mode\_requested - Lock mode requested monitor element"* on page 913  
*"lock\_name - Lock name monitor element"* on page 914  
*"lock\_object\_type - Lock object type waited on monitor element"* on page 916  
*"lock\_release\_flags - Lock release flags monitor element"* on page 918  
*"lock\_wait\_start\_time - Lock wait start timestamp monitor element"* on page 923  
*"node\_number - Node Number"* on page 970

- “ss\_number - Subsection number monitor element” on page 1238
- “table\_name - Table name monitor element” on page 1269
- “table\_schema - Table schema name monitor element” on page 1271
- “tablespace\_name - Table space name monitor element” on page 1279

### **memory\_pool logical data group**

- “node\_number - Node Number” on page 970
- “pool\_config\_size - Configured Size of Memory Pool” on page 1040
- “pool\_cur\_size - Current Size of Memory Pool” on page 1041
- “pool\_id - Memory Pool Identifier” on page 1071
- “pool\_secondary\_id - Memory Pool Secondary Identifier” on page 1114
- “pool\_watermark - Memory Pool Watermark” on page 1127

### **progress logical data group**

- “progress\_completed\_units - Completed Progress Work Units” on page 1159
- “progress\_description - Progress Description” on page 1160
- “progress\_seq\_num - Progress Sequence Number” on page 1160
- “progress\_start\_time - Progress Start Time” on page 1161
- “progress\_total\_units - Total Progress Work Units” on page 1161
- “progress\_work\_metric - Progress Work Metric” on page 1161

### **progress\_list logical data group**

- “progress\_list\_attr - Current Progress List Attributes” on page 1160
- “progress\_list\_cur\_seq\_num - Current Progress List Sequence Number” on page 1160

### **rollforward logical data group**

- “node\_number - Node Number” on page 970
- “rf\_log\_num - Log being rolled forward monitor element” on page 1183
- “rf\_status - Log Phase” on page 1184
- “rf\_timestamp - Rollforward Timestamp” on page 1184
- “rf\_type - Rollforward Type” on page 1184
- “ts\_name - Table space being rolled forward monitor element” on page 1397

### **stmt logical data group**

- “agents\_top - Number of Agents Created” on page 658
- “blocking\_cursor - Blocking Cursor” on page 691
- “consistency\_token - Package consistency token monitor element” on page 738
- “creator - Application Creator” on page 763
- “cursor\_name - Cursor Name” on page 766
- “degree\_parallelism - Degree of Parallelism” on page 785
- “fetch\_count - Number of Successful Fetches” on page 847
- “int\_rows\_deleted - Internal Rows Deleted” on page 884
- “int\_rows\_inserted - Internal Rows Inserted” on page 884
- “int\_rows\_updated - Internal Rows Updated” on page 885
- “num\_agents - Number of Agents Working on a Statement” on page 971
- “package\_name - Package name monitor element” on page 1006

"package\_version\_id - Package version monitor element" on page 1007  
"pool\_data\_l\_reads - Buffer pool data logical reads monitor element" on page 1048  
"pool\_data\_p\_reads - Buffer pool data physical reads monitor element" on page 1050  
"pool\_index\_l\_reads - Buffer pool index logical reads monitor element" on page 1080  
"pool\_index\_p\_reads - Buffer pool index physical reads monitor element" on page 1082  
"pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element" on page 1116  
"pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element" on page 1118  
"pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element" on page 1120  
"pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element" on page 1121  
"pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element" on page 1123  
"pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element" on page 1125  
"pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element" on page 1136  
"pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element" on page 1139  
"query\_card\_estimate - Query Number of Rows Estimate" on page 1163  
"query\_cost\_estimate - Query cost estimate monitor element" on page 1163  
"rows\_read - Rows read monitor element" on page 1191  
"rows\_written - Rows Written" on page 1196  
"section\_number - Section number monitor element" on page 1201  
"sort\_overflows - Sort overflows monitor element" on page 1224  
"stmt\_elapsed\_time - Most Recent Statement Elapsed Time" on page 1245  
"stmt\_node\_number - Statement Node" on page 1250  
"stmt\_operation/operation - Statement operation monitor element" on page 1250  
"stmt\_sorts - Statement Sorts" on page 1253  
"stmt\_start - Statement Operation Start Timestamp" on page 1254  
"stmt\_stop - Statement Operation Stop Timestamp" on page 1254  
"stmt\_sys\_cpu\_time - System CPU Time used by Statement" on page 1255  
"stmt\_text - SQL statement text monitor element" on page 1255  
"stmt\_type - Statement type monitor element" on page 1256  
"stmt\_usr\_cpu\_time - User CPU Time used by Statement" on page 1258  
"total\_sort\_time - Total sort time monitor element" on page 1374

### **stmt\_transmissions logical data group**

"elapsed\_exec\_time - Statement Execution Elapsed Time" on page 806  
"host\_response\_time - Host Response Time" on page 871

*“max\_data\_received\_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes” on page 945*

*“max\_data\_received\_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes” on page 946*

*“max\_data\_received\_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes” on page 946*

*“max\_data\_received\_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes” on page 947*

*“max\_data\_received\_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes” on page 947*

*“max\_data\_received\_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element” on page 948*

*“max\_data\_received\_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes” on page 948*

*“max\_data\_received\_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes” on page 948*

*“max\_data\_received\_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element” on page 949*

*“max\_data\_received\_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes” on page 949*

*“max\_data\_received\_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes” on page 950*

*“max\_data\_sent\_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes” on page 950*

*“max\_data\_sent\_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes” on page 950*

*“max\_data\_sent\_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes” on page 951*

*“max\_data\_sent\_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes” on page 951*

*“max\_data\_sent\_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes” on page 952*

*“max\_data\_sent\_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes” on page 952*

*“max\_data\_sent\_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes” on page 953*

*“max\_data\_sent\_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes” on page 953*

*“max\_data\_sent\_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes” on page 954*

*“max\_data\_sent\_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes” on page 954*

*“max\_data\_sent\_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes” on page 954*

*“max\_network\_time\_100\_ms - Number of Statements with Network Time between 16 and 100 ms” on page 955*

*“max\_network\_time\_16\_ms - Number of Statements with Network Time between 4 and 16 ms” on page 955*

*“max\_network\_time\_1\_ms - Number of Statements with Network Time of up to 1 ms” on page 956*

"max\_network\_time\_4\_ms - Number of Statements with Network Time between 1 and 4 ms" on page 956  
 "max\_network\_time\_500\_ms - Number of Statements with Network Time between 100 and 500 ms" on page 957  
 "max\_network\_time\_gt500\_ms - Number of Statements with Network Time greater than 500 ms" on page 957  
 "network\_time\_bottom - Minimum Network Time for Statement" on page 968  
 "network\_time\_top - Maximum Network Time for Statement" on page 969  
 "outbound\_bytes\_received - Outbound Number of Bytes Received" on page 1001  
 "outbound\_bytes\_received\_bottom - Minimum Outbound Number of Bytes Received" on page 1002  
 "outbound\_bytes\_received\_top - Maximum Outbound Number of Bytes Received" on page 1002  
 "outbound\_bytes\_sent - Outbound Number of Bytes Sent" on page 1002  
 "outbound\_bytes\_sent\_bottom - Minimum Outbound Number of Bytes Sent" on page 1003  
 "outbound\_bytes\_sent\_top - Maximum Outbound Number of Bytes Sent" on page 1003  
 "sql\_chains - Number of SQL Chains Attempted" on page 1228  
 "sql\_stmts - Number of SQL Statements Attempted" on page 1229

### **subsection logical data group**

"rows\_read - Rows read monitor element" on page 1191  
 "rows\_written - Rows Written" on page 1196  
 "ss\_exec\_time - Subsection Execution Elapsed Time" on page 1237  
 "ss\_node\_number - Subsection Node Number" on page 1238  
 "ss\_number - Subsection number monitor element" on page 1238  
 "ss\_status - Subsection status monitor element" on page 1238  
 "ss\_sys\_cpu\_time - System CPU Time used by Subsection" on page 1239  
 "ss\_usr\_cpu\_time - User CPU Time used by Subsection" on page 1239  
 "tq\_cur\_send\_spills - Current number of table queue buffers overflowed monitor element" on page 1389  
 "tq\_id\_waiting\_on - Waited on node on a table queue monitor element" on page 1390  
 "tq\_max\_send\_spills - Maximum number of table queue buffers overflows" on page 1390  
 "tq\_node\_waited\_for - Waited for node on a table queue" on page 1391  
 "tq\_rows\_read - Number of Rows Read from table queues" on page 1391  
 "tq\_rows\_written - Number of rows written to table queues" on page 1391  
 "tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element" on page 1396  
 "tq\_wait\_for\_any - Waiting for any node to send on a table queue" on page 1397

### **table logical data group**

"data\_object\_pages - Data Object Pages" on page 766  
 "data\_partition\_id - Data partition identifier monitor element" on page 767  
 "index\_object\_pages - Index Object Pages" on page 875

"lob\_object\_pages - LOB Object Pages" on page 898  
"long\_object\_pages - Long Object Pages" on page 941  
"overflow\_accesses - Accesses to overflowed records monitor element" on page 1004  
"page\_reorgs - Page reorganizations monitor element" on page 1009  
"rows\_read - Rows read monitor element" on page 1191  
"rows\_written - Rows Written" on page 1196  
"table\_file\_id - Table file ID monitor element" on page 1268  
"table\_name - Table name monitor element" on page 1269  
"table\_schema - Table schema name monitor element" on page 1271  
"table\_type - Table type monitor element" on page 1273  
"tablespace\_id - Table space identification monitor element" on page 1276  
"xda\_object\_pages - XDA Object Pages" on page 1430

### **table\_list logical data group**

"db\_conn\_time - Database activation timestamp monitor element" on page 772  
"db\_name - Database name monitor element" on page 773  
"db\_path - Database Path" on page 774  
"input\_db\_alias - Input Database Alias" on page 877  
"last\_reset - Last Reset Timestamp" on page 896

### **table\_reorg logical data group**

"data\_partition\_id - Data partition identifier monitor element" on page 767  
"reorg\_completion - Reorganization Completion Flag" on page 1176  
"reorg\_current\_counter - Reorganize Progress" on page 1176  
"reorg\_end - Table Reorganize End Time" on page 1177  
"reorg\_index\_id - Index Used to Reorganize the Table" on page 1177  
"reorg\_max\_counter - Total Amount of Reorganization" on page 1177  
"reorg\_max\_phase - Maximum Reorganize Phase" on page 1178  
"reorg\_phase - Table reorganization phase monitor element" on page 1178  
"reorg\_phase\_start - Reorganize Phase Start Time" on page 1179  
"reorg\_rows\_compressed - Rows Compressed" on page 1179  
"reorg\_rows\_rejected\_for\_compression - Rows Rejected for Compression" on page 1179  
"reorg\_start - Table Reorganize Start Time" on page 1179  
"reorg\_status - Table Reorganize Status" on page 1180  
"reorg\_tbsp\_id - Table Space Where Table or Data partition is Reorganized" on page 1180  
"reorg\_type - Table Reorganize Attributes" on page 1180  
"reorg\_xml\_regions\_compressed - XML regions compressed monitor element" on page 1181  
"reorg\_xml\_regions\_rejected\_for\_compression - XML regions rejected for compression monitor element" on page 1181

### **tablespace logical data group**

"direct\_read\_reqs - Direct read requests monitor element" on page 790  
"direct\_read\_time - Direct read time monitor element" on page 792

“direct\_reads - Direct reads from database monitor element” on page 794

“direct\_write\_reqs - Direct write requests monitor element” on page 796

“direct\_write\_time - Direct write time monitor element” on page 798

“direct\_writes - Direct writes to database monitor element” on page 800

“files\_closed - Database files closed monitor element” on page 848

“fs\_caching - File system caching monitor element” on page 849

“pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element” on page 1026

“pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element” on page 1027

“pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element” on page 1028

“pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element” on page 1031

“pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element” on page 1032

“pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element” on page 1033

“pool\_async\_read\_time - Buffer Pool Asynchronous Read Time” on page 1033

“pool\_async\_write\_time - Buffer pool asynchronous write time monitor element” on page 1034

“pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element” on page 1038

“pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element” on page 1038

“pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element” on page 1039

“pool\_data\_l\_reads - Buffer pool data logical reads monitor element” on page 1048

“pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050

“pool\_data\_writes - Buffer pool data writes monitor element” on page 1052

“pool\_index\_l\_reads - Buffer pool index logical reads monitor element” on page 1080

“pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082

“pool\_index\_writes - Buffer pool index writes monitor element” on page 1084

“pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element” on page 1087

“pool\_read\_time - Total buffer pool physical read time monitor element” on page 1113

“pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element” on page 1116

“pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element” on page 1118

“pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element” on page 1120

“pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element” on page 1121

“pool\_temp\_xda\_l\_reads - Buffer pool temporary XDA data logical reads monitor element” on page 1123

“pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element” on page 1125

“pool\_write\_time - Total buffer pool physical write time monitor element” on page 1128

“pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element” on page 1136

“pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

“pool\_xda\_writes - Buffer pool XDA data writes monitor element” on page 1141

“tablespace\_auto\_resize\_enabled - Table space automatic resizing enabled monitor element” on page 1273

“tablespace\_content\_type - Table space content type monitor element” on page 1274

“tablespace\_cur\_pool\_id - Buffer pool currently being used monitor element” on page 1274

“tablespace\_extent\_size - Table space extent size monitor element” on page 1275

“tablespace\_id - Table space identification monitor element” on page 1276

“tablespace\_name - Table space name monitor element” on page 1279

“tablespace\_next\_pool\_id - Buffer pool that will be used at next startup monitor element” on page 1280

“tablespace\_page\_size - Table space page size monitor element” on page 1281

“tablespace\_prefetch\_size - Table space prefetch size monitor element” on page 1283

“tablespace\_rebalancer\_mode - Rebalancer mode monitor element” on page 1285

“tablespace\_type - Table space type monitor element” on page 1292

“tablespace\_using\_auto\_storage - Table space enabled for automatic storage monitor element” on page 1294

### **tablespace\_container logical data group**

“container\_accessible - Accessibility of container monitor element” on page 738

“container\_id - Container identification monitor element” on page 739

“container\_name - Container name monitor element” on page 739

“container\_stripe\_set - Container stripe set monitor element” on page 739

“container\_total\_pages - Total pages in container monitor element” on page 740

“container\_type - Container type monitor element” on page 740

“container\_usable\_pages - Usable pages in container monitor element” on page 741

### **tablespace\_list logical data group**

“db\_conn\_time - Database activation timestamp monitor element” on page 772

“db\_name - Database name monitor element” on page 773

“db\_path - Database Path” on page 774

“input\_db\_alias - Input Database Alias” on page 877

“last\_reset - Last Reset Timestamp” on page 896

## **tablespace\_nodeinfo logical data group**

- “tablespace\_current\_size - Current table space size” on page 1275*
- “tablespace\_free\_pages - Free pages in table space monitor element” on page 1275*
- “tablespace\_increase\_size - Increase size in bytes” on page 1277*
- “tablespace\_increase\_size\_percent - Increase size by percent monitor element” on page 1277*
- “tablespace\_initial\_size - Initial table space size” on page 1277*
- “tablespace\_last\_resize\_failed - Last resize attempt failed” on page 1278*
- “tablespace\_last\_resize\_time - Time of last successful resize” on page 1278*
- “tablespace\_max\_size - Maximum table space size” on page 1278*
- “tablespace\_min\_recovery\_time - Minimum recovery time for rollforward monitor element” on page 1279*
- “tablespace\_num\_containers - Number of Containers in Table Space” on page 1281*
- “tablespace\_num\_quiescers - Number of Quiescers” on page 1281*
- “tablespace\_num\_ranges - Number of Ranges in the Table Space Map” on page 1281*
- “tablespace\_page\_top - Table space high watermark monitor element” on page 1282*
- “tablespace\_paths\_dropped - Table space using dropped path monitor element” on page 1282*
- “tablespace\_pending\_free\_pages - Pending free pages in table space monitor element” on page 1283*
- “tablespace\_prefetch\_size - Table space prefetch size monitor element” on page 1283*
- “tablespace\_rebalancer\_extents\_processed - Number of extents the rebalancer has processed” on page 1284*
- “tablespace\_rebalancer\_extents\_remaining - Total number of extents to be processed by the rebalancer” on page 1284*
- “tablespace\_rebalancer\_last\_extent\_moved - Last extent moved by the rebalancer” on page 1285*
- “tablespace\_rebalancer\_priority - Current rebalancer priority” on page 1286*
- “tablespace\_rebalancer\_restart\_time - Rebalancer restart time” on page 1287*
- “tablespace\_rebalancer\_start\_time - Rebalancer start time” on page 1288*
- “tablespace\_state - Table space state monitor element” on page 1289*
- “tablespace\_state\_change\_object\_id - State Change Object Identification” on page 1291*
- “tablespace\_state\_change\_ts\_id - State Change Table Space Identification” on page 1291*
- “tablespace\_total\_pages - Total pages in table space monitor element” on page 1292*
- “tablespace\_usable\_pages - Usable pages in table space monitor element” on page 1293*
- “tablespace\_used\_pages - Used pages in table space monitor element” on page 1293*

### **tablespace\_quiescer logical data group**

- “quiescer\_agent\_id - Quiescer Agent Identification” on page 1166
- “quiescer\_auth\_id - Quiescer User Authorization Identification” on page 1167
- “quiescer\_obj\_id - Quiescer Object Identification” on page 1167
- “quiescer\_state - Quiescer State” on page 1167
- “quiescer\_ts\_id - Quiescer Table Space Identification” on page 1168

### **tablespace\_range logical data group**

- “range\_adjustment - Range Adjustment” on page 1168
- “range\_container\_id - Range Container” on page 1168
- “range\_end\_stripe - End Stripe” on page 1168
- “range\_max\_extent - Maximum Extent in Range” on page 1169
- “range\_max\_page\_number - Maximum Page in Range” on page 1169
- “range\_num\_containers - Number of Containers in Range” on page 1169
- “range\_number - Range Number” on page 1169
- “range\_offset - Range Offset” on page 1169
- “range\_start\_stripe - Start Stripe” on page 1170
- “range\_stripe\_set\_number - Stripe Set Number” on page 1170

### **utility\_info logical data group**

- “node\_number - Node Number” on page 970
- “utility\_dbname - Database Operated on by Utility” on page 1411
- “utility\_description - Utility Description” on page 1412
- “utility\_id - Utility ID” on page 1412
- “utility\_invoker\_type - Utility Invoker Type” on page 1413
- “utility\_priority - Utility Priority” on page 1415
- “utility\_start\_time - Utility Start Time” on page 1416
- “utility\_state - Utility State” on page 1416
- “utility\_type - Utility Type” on page 1417

---

## Chapter 11. Monitor element reference

A description of the data collected by the monitor element.

The monitor elements returned by the system monitor fall into the following categories:

- **Identification** for the database manager, an application, or a database connection being monitored.
- Data primarily intended to help you to **configure** the system.
- Database **activity** at various levels including database, application, table, or statement. This information can be used for activity monitoring, problem determination, and performance analysis. It can also be used for configuration.
- information about **DB2 Connect** applications. Including information about DCS applications running at the gateway, SQL statements being executed, and database connections.
- information about **Federated Database Systems**. This includes information about the total access to a data source by applications running in a DB2 federated system and information about access to a data source by a given application running in a federated server instance.

Monitor elements are described in a standard format as follows:

### Element identifier

The name of the element. If parsing the data stream directly, the element identifier is uppercase and prefixed with SQLM\_ELM\_.

### Element type

The type of information the monitor element returns. For example, the db2start\_time monitor element returns a timestamp.

### Table function monitoring information

If a monitor element is returned by a table function, a table with the following fields is shown.

- *Table Function*: The name of the table function that returns the monitor element.
- *Monitor Element Collection Level*: For more information about monitor element collection levels, see Chapter 8, “Monitor element collection levels,” on page 525.

### Snapshot monitoring information

If a monitor element returns snapshot monitoring information, a table with the following fields is shown.

- *Snapshot level*: The level of information that can be captured by the snapshot monitor. For example, the appl\_status monitor element returns information at the Application level and at the Lock level.
- *Logical data grouping*: The logical data group where captured snapshot information is returned. If parsing the data stream directly, the logical data group identifier is uppercase and prefixed with SQLM\_ELM\_. For example, the appl\_status monitor element returns information for the appl\_id\_info grouping and for the appl\_lock\_list grouping.
- *Monitor switch*: The system monitor switch that must be set to obtain this information. If the switch is Basic, data will always be collected for the monitor element.

### Event monitoring information

If a monitor element is collected by event monitors, a table with the following fields is shown.

- *Event type*: The level of information that can be collected by the event monitor. The event monitor must be created with this event type to collect this information. For example, the `appl_status` monitor element is collected for CONNECTIONS event monitors.
- *Logical data grouping*: The logical data group where captured event information is returned. If parsing the data stream directly, the logical data group identifier is uppercase and prefixed with `SQLM_ELM_`. For example, the `appl_status` monitor element returns information for the `event_conn` grouping.
- *Monitor switch*: The system monitor switch that must be set to obtain this information. For event monitors, the `TIMESTAMP` switch is the only monitor switch that can restrict the collection of event data. If there is a dash shown for this field, data will always be collected for the monitor element.

**Usage** information about how you can use the information collected by the monitor element when monitoring your database system.

---

## acc\_curs\_blk - Accepted Block Cursor Requests

The number of times that a request for an I/O block was accepted.

### Element identifier

`acc_curs_blk`

### Element type

counter

Table 150. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 151. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** You can use this element in conjunction with `rej_curs_blk` to calculate the percentage of blocking requests that are accepted, rejected, or both.

See `rej_curs_blk` for suggestions on how to use this information to tune your configuration parameters.

---

## act\_aborted\_total - Total aborted activities monitor element

The total number of coordinator activities at any nesting level that completed with errors. For service classes, if an activity is remapped to a different service subclass with a `REMAP ACTIVITY` action before it aborts, then this activity counts only toward the total of the subclass it aborts in.

Table 152. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 153. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## Usage

Use this element to understand if activities on the system are completing successfully. Activities may be aborted due to cancellation, errors, or reactive thresholds.

### **act\_completed\_total - Total completed activities monitor element**

The total number of coordinator activities at any nesting level that completed successfully. For service classes, if an activity is remapped to a different subclass with a REMAP ACTIVITY action before it completes, then this activity counts only toward the total of the subclass it completes in.

Table 154. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics  | REQUEST METRICS BASE   |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample   | REQUEST METRICS BASE   |

Table 155. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## Usage

Use this element to determine the throughput of activities in the system.

## act\_cpu\_time\_top – Activity CPU time top monitor element

The high watermark for processor time used by activities at all nesting levels in a service class, workload, or work class. This value is reported in microseconds.

The monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class or workload in which the activity runs is set to NONE. Activities contribute toward this high watermark only when request metrics are enabled. If the collection of activity metrics is not enabled, a value of 0 is returned.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action, only the act\_cpu\_time\_top high watermark of the service subclass where an activity completes is updated, provided that a new high watermark is reached. The act\_cpu\_time\_top high watermarks of other service subclasses an activity is mapped to but does not complete in are unaffected.

Table 156. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_scstats         | Always collected |
| Statistics | event_wcstats         | Always collected |
| Statistics | event_wlstats         | Always collected |

## Usage

Use this element to determine the highest amount of processor time used by an activity on a member for a service class, workload, or work class during the time interval collected.

---

## act\_exec\_time - Activity execution time monitor element

Time spent executing at this member, in microseconds. For cursors, the execution time is the combined time for the open, the fetches, and the close. The time when the cursor is idle is not counted toward execution time. For routines, execution time is the start to end of routine invocation. The lifetimes of any cursors left open by routine (to return a result set) after the routine finishes are not counted toward the routine execution time. For all other activities, execution time is the difference between start time and stop time. In all cases, execution time does not include time spent initializing or queued.

Table 157. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

## Usage

This element can be used alone to know the elapsed time spent executing the activity by DB2 on each member. This element can also be used together with **time\_started** and **time\_completed** monitor elements on the coordinator member to compute the idle time for cursor activities. You can use the following formula:

Cursor idle time = (time\_completed - time\_started) - act\_exec\_time

---

## act\_rejected\_total - Total rejected activities monitor element

The total number of coordinator activities at any nesting level that were rejected instead of being allowed to execute.

*Table 158. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 159. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

### Usage

Use this element to help determine whether predictive thresholds and work actions that prevent execution are effective and whether they are too restrictive.

---

## act\_remapped\_in – Activities remapped in monitor element

Count of the number of activities to be remapped into this service subclass since the last reset.

Table 160. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

### Usage

Use this count to determine whether the remapping of activities into the service subclass is occurring as desired.

---

## act\_remapped\_out – Activities remapped out monitor element

Count of the number of activities to be remapped out of this service subclass since the last reset.

Table 161. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

### Usage

Use this count to determine whether the remapping of activities out of the service subclass is occurring as desired.

---

## act\_rows\_read\_top – Activity rows read top monitor element

The high watermark for the number of rows read by activities at all nesting levels in a service class, workload, or work class.

The monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class or workload in which the activity runs is set to NONE. Activities contribute toward this high watermark only when request metrics are enabled. If the collection of activity metrics is not enabled, a value of 0 is returned.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action only the act\_rows\_read\_top high watermark of the service subclass where an activity completes is updated, provided that a new high watermark is reached. The act\_rows\_read\_top high watermarks of service subclasses an activity is mapped to but does not complete in are unaffected.

Table 162. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_scstats         | Always collected |
| Statistics | event_wcstats         | Always collected |
| Statistics | event_wlstats         | Always collected |

## Usage

Use this element to determine the highest number of rows read by an activity on a member for a service class, workload, or work class during the time interval collected.

---

### act\_rqsts\_total - Total activity requests monitor elements

The number of individual coordinator and subagent requests completed as part of an activity. For example, a fetch on a cursor activity.

*Table 163. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 164. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

---

## act\_throughput - Activity throughput monitor element

The rate at which coordinator activities are completed at any nesting level. Measured in coordinator activities per second.

Table 165. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics                        | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample workload metrics                                  | REQUEST METRICS BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service<br>subclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                         | ACTIVITY METRICS BASE            |

Table 166. Event Monitoring Information

| Event Type | Logical Data Grouping                                   | Monitor Switch   |
|------------|---|------------------|
| Statistics | event_scstats (reported in the<br>details_xml document) | Always collected |
| Statistics | event_wlstats (reported in the<br>details_xml document) | Always collected |

### Usage

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS function, this monitor element represents the activity throughput since the last reset of the statistics.

When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS function, this monitor element represents the activity throughput since the function was executed.

---

## act\_total - Activities total monitor element

Total number of activities at any nesting level that had work actions corresponding to the specified work class applied to them since the last reset.

Table 167. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_WORK_ACTION_SET_STATS<br>table function - Return work action set<br>statistics | ACTIVITY METRICS BASE            |

Table 168. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wcstats         | -              |

## Usage

Every time an activity has one or more work actions associated with a work class applied to it, a counter for the work class is updated. This counter is exposed using the **act\_total** monitor element. The counter can be used to judge the effectiveness of the work action set (for example, how many activities have a actions applied). It can also be used to understand the different types of activities on the system.

---

## activate\_timestamp - Activate timestamp monitor element

The time when an event monitor was activated.

*Table 169. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Activity             | event_activity            | -              |
| Activity             | event_activitystmt        | -              |
| Activity             | event_activityvals        | -              |
| Threshold Violations | event_thresholdviolations | -              |

## Usage

Use this element to correlate information returned by the previously mentioned event types.

---

## active\_hash\_joins - Active hash joins

The total number of hash joins that are currently running and consuming memory.

*Table 170. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

---

## active\_olap\_funcs - Active OLAP Functions monitor element

The total number of OLAP functions that are currently running and consuming sort heap memory.

*Table 171. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

For snapshot monitoring, this counter can be reset.

---

## active\_sorts - Active Sorts

The number of sorts in the database that currently have a sort heap allocated.

*Table 172. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** Use this value in conjunction with *sort\_heap\_allocated* to determine the average sort heap space used by each sort. If the *sortheap* configuration parameter is substantially larger than the average sort heap used, you may be able to lower the value of this parameter.

This value includes heaps for sorts of temporary tables that were created during relational operations.

---

## activity\_collected - Activity collected monitor element

This element indicates whether or not activity event monitor records are to be collected for a violated threshold.

*Table 173. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

### Usage

Use this element to determine whether to expect an activity event for the activity that violated the threshold to be written to the activity event monitor.

When an activity finishes or aborts and the activity event monitor is active at the time, if the value of this monitor element is 'Y', the activity that violated this threshold will be collected. If the value of this monitor element is 'N', it will not be collected.

---

## activity\_id - Activity ID monitor element

Counter which uniquely identifies an activity for an application within a given unit of work.

*Table 174. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                       | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES table function - Return a list of activities                       | ACTIVITY METRICS BASE            |

*Table 175. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Locking    | -                     | Always collected |
| Activities | event_activity        | Always collected |
| Activities | event_activitystmt    | Always collected |
| Activities | event_activityvals    | Always collected |

Table 175. Event Monitoring Information (continued)

| Event Type           | Logical Data Grouping     | Monitor Switch        |
|----------------------|---------------------------|-----------------------|
| Activities           | event_activitymetrics     | ACTIVITY METRICS BASE |
| Threshold violations | event_thresholdviolations | Always collected      |

## Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity.

To uniquely identify an activity outside its unit of work, use the combination of **activity\_id** and **uow\_id** plus one of the following monitor elements: **appl\_id** or **agent\_id**.

---

## activity\_secondary\_id - Activity secondary ID monitor element

The value for this element is incremented each time an activity record is written for the same activity. For example, if an activity record is written once as a result of having called the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS procedure and a second time when the activity ends, the element would have a value of 0 for the first record and 1 for the second record.

Table 176. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch        |
|------------|-----------------------|-----------------------|
| Activities | event_activity        | -                     |
| Activities | event_activitystmt    | -                     |
| Activities | event_activityvals    | -                     |
| Activities | event_activitymetrics | ACTIVITY METRICS BASE |

## Usage

Use this element with **activity\_id**, **uow\_id**, and **appl\_id** monitor elements to uniquely identify activity records when information about the same activity has been written to the activities event monitor multiple times.

For example, information about an activity would be sent to the activities event monitor twice in the following case:

- the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS stored procedure was used to capture information about the activity while it was running
- information about the activity was collected when the activity completed, because the COLLECT ACTIVITY DATA clause was specified on the service class with which the activity is associated

---

## activity\_state - Activity state monitor element

The current state of the activity.

Table 177. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE                        |

### Usage

Use this monitor element to understand what the activity is currently doing (for example, is the activity stuck in a queue or waiting for input from the client). Possible values include:

- CANCEL\_PENDING
- EXECUTING
- IDLE
- INITIALIZING
- QP\_CANCEL\_PENDING
- QP\_QUEUED
- QUEUED
- TERMINATING
- UNKNOWN

---

## activity\_type - Activity type monitor element

The type of the activity.

Table 178. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 179. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

The possible values are:

- LOAD
- READ\_DML
- WRITE\_DML

- DDL
- CALL
- OTHER

The value OTHER is returned for SET statements that do not perform SQL (for example, SET special register, or SET EVENT MONITOR STATE) and the LOCK TABLE statement.

---

## activitytotaltime\_threshold\_id - Activity total time threshold ID monitor element

The ID of the ACTIVITYTOTALTIME threshold that was applied to the activity.

*Table 180. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

### Usage

Use this element to understand which ACTIVITYTOTALTIME threshold, if any, was applied to the activity.

---

## activitytotaltime\_threshold\_value - Activity total time threshold value monitor element

A timestamp that is computed by adding the ACTIVITYTOTALTIME threshold duration to the activity entry time. If the activity is still executing when this timestamp is reached, the threshold will be violated.

*Table 181. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

### Usage

Use this element to understand the value of the ACTIVITYTOTALTIME threshold applied to the activity, if any.

---

## activitytotaltime\_threshold\_violated - Activity total time threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the ACTIVITYTOTALTIME threshold. 'No' indicates that the activity has not yet violated the threshold.

Table 182. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

### Usage

Use this element to determine if the activity violated the ACTIVITYTOTALTIME threshold that was applied to the activity.

---

## adapter\_name - Adapter name monitor element

Name of the network adapter on this host.

Table 183. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

## address - IP address from which the connection was initiated

The IP address from which the activity connection was initiated.

Table 184. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences | ACTIVITY METRICS BASE            |

Table 185. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

### Usage

Use this to identify the IP address from which the activity connection was initiated. Secure domain names are shown converted to an IP address.

## agent\_id - Application handle (agent ID) monitor element

A system-wide unique ID for the application. In a single-member database configuration, this identifier consists of a 16-bit counter. In a multi-member configuration, this identifier consists of the coordinating member number concatenated with a 16-bit counter. In addition, this identifier is the same on every member where the application may make a secondary connection.

Table 186. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database                                  | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 187. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcs_appl_info         | Basic          |
| Transaction     | event_xact            | -              |

Table 188. Event Monitoring Information

| Event Type                          | Logical Data Grouping     | Monitor Element Collection Level |
|-------------------------------------|---------------------------|----------------------------------|
| Locking                             | -                         | Always collected                 |
| Unit of work                        | -                         | Always collected                 |
| Connections                         | event_connheader          | Always collected                 |
| Statements                          | event_stmt                | Always collected                 |
| Statements                          | event_subsection          | Always collected                 |
| Deadlocks <sup>1</sup>              | event_dlconn              | Always collected                 |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn     | Always collected                 |
| Threshold violations                | event_thresholdviolations | Always collected                 |
| Activities                          | event_activity            | Always collected                 |

Table 188. Event Monitoring Information (continued)

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Change history | changesummary         | Always collected                 |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The application handle, also known as the agent ID, can be used to uniquely identify an active application.

**Note:** The **agent\_id** monitor element has different behavior depending on your version of DB2. When taking snapshots from DB2 with version SQLM\_DBMON\_VERSION1 or SQLM\_DBMON\_VERSION2 to a DB2 (Version 5 or greater) database, the **agent\_id** returned is not usable as an application identifier, rather it is the **agent\_pid** of the agent serving the application. In these cases an **agent\_id** is still returned for compatibility with earlier releases, but internally the DB2 database server will not recognize the value as an **agent\_id**.

This value can be used as input to GET SNAPSHOT commands that require an agent ID or to the monitor table functions that require an application handle.

When reading event traces, it can be used to match event records with a given application.

It can also be used as input to the FORCE APPLICATION command or API. On multi-node systems this command can be issued from any node where the application has a connection. Its effect is global.

---

## agent\_id\_holding\_lock - Agent ID Holding Lock

The application handle of the agent holding a lock for which this application is waiting. The lock monitor group must be turned on to obtain this information.

Table 189. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock_wait             | Lock           |

**Usage** This element can help you determine which applications are in contention for resources.

If this element is 0 (zero) and the application is waiting for a lock, this indicates that the lock is held by an indoubt transaction. You can use either `appl_id_holding_lk` or the command line processor LIST INDOUBT TRANSACTIONS command (which displays the application ID of the CICS® agent that was processing the transaction when it became indoubt) to determine the indoubt transaction, and then either commit it or roll it back.

Note that more than one application can hold a shared lock on an object for which this application is waiting. See `lock_mode` for information about the type of lock that the application holds. If you are taking an application snapshot, only one of the agent IDs holding a lock on the object will be returned. If you are taking a lock snapshot, all of the agent IDs holding a lock on the object will be identified.

---

## agent\_pid - Engine dispatchable unit (EDU) identifier monitor element

The unique identifier for the engine dispatchable unit (EDU) for the agent. Except on the Linux operating system, the EDU ID is mapped to the thread ID. On the Linux operating system, the EDU ID is a DB2 generated unique identifier.

*Table 190. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | agent                 | Statement      |

*Table 191. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | -                     | -              |

### Usage

You can use this element to link database system monitor information to other sources of diagnostic information, such as system traces. You can also use it to monitor how agents working for a database application use system resources.

---

## agent\_status - DCS Application Agents

In a connection concentrator environment, this value shows which applications currently have associated agents.

### Element identifier

agent\_status

### Element type

information

*Table 192. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcx_appl_info         | Basic          |

### Usage Values are:

- `SQLM_AGENT_ASSOCIATED`  
The agent working on behalf of this application is associated with it.
- `SQLM_AGENT_NOT_ASSOCIATED`  
The agent that was working on behalf of this application is no longer associated with it and is being used by another application. The next time work is done for this application without an associated agent, an agent will be re-associated.

---

## agent\_sys\_cpu\_time - System CPU Time used by Agent

The total *system* CPU time (in seconds and microseconds) used by the database manager agent process.

**Element identifier**

agent\_sys\_cpu\_time

**Element type**

time

*Table 193. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Timestamp      |

For snapshot monitoring at the application level, this counter can be reset. This counter cannot be reset at other levels.

**Usage** This element along with the other related CPU-time elements can help you understand the level of activity within an application, and may help you identify applications that could benefit from additional tuning.

This element includes CPU time for both SQL and non-SQL statements, as well as CPU time for any unfenced user-defined functions (UDFs)

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

**Note:** If this information is not available for your operating system, this element will be set to 0.

---

## agent\_tid - Agent thread ID monitor element

The thread ID of the agent or system entity. If this ID is unavailable, the value of the column is null.

*Table 194. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function - Get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                   | ACTIVITY METRICS BASE            |

*Table 195. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## agent\_usr\_cpu\_time - User CPU Time used by Agent

The total CPU time (in seconds and microseconds) used by database manager agent process.

**Element identifier**  
agent\_usr\_cpu\_time

**Element type**  
time

Table 196. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Timestamp      |

For snapshot monitoring, this counter can be reset.

**Usage** This element along with the other CPU-time related elements can help you identify applications or queries that consume large amounts of CPU.

This counter includes time spent on both SQL and non-SQL statements, as well as any unfenced user-defined functions (UDFs) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

**Note:** If this information is not available for your operating system, this element will be returned as 0.

---

## agent\_wait\_time - Agent wait time monitor element

Time spent by an application queued to wait for an agent under concentrator configurations. The value is given in milliseconds.

Table 197. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 197. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)      | REQUEST METRICS BASE             |

Table 198. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | Always collected     |

## Usage

The **agent\_wait\_time** monitor element can be used to help evaluate how efficiently your system is running in a concentrator environment. A high agent wait relative to the **total\_request\_time** monitor element value indicates that requests are spending a lot of time queued waiting for agents, which may be indicative of one or more of the following events:

- The **max\_coordagents** configuration parameter has been configured too small for your workload. You may need to increase the value of **max\_coordagents** configuration parameter, or the ratio of **max\_coordagents** configuration parameter to **max\_connections** configuration parameter if you are running with both parameters set to AUTOMATIC, to ensure that enough coordinator agents are available to service your application requests in a timely manner.
- Your workload is not committing frequently enough. For the concentrator to work efficiently, applications should issue commits relatively frequently to ensure that their agents can be freed up to serve requests on other applications. If your applications do not do frequent commits you may need to configure a proportionally higher number of coordinator agents to reduce the time spent waiting for agents to become available.

## agent\_waits\_total - Total agent waits monitor element

Number of times an application had to wait for an agent to be assigned under concentrator configurations.

Table 199. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 199. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 200. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## Usage

Use this element in conjunction with the **agent\_wait\_time** monitor element to determine the average amount of time an application request spends waiting for an agent in a concentrator environment.

## agents\_created\_empty\_pool - Agents Created Due to Empty Agent Pool

The number of agents created because the agent pool was empty. It includes the number of agents started at **db2start** time. (*num\_initagents*).

Table 201. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

**Usage** In conjunction with `agents_from_pool`, you can calculate the ratio of  
Agents Created Due to Empty Agent Pool / Agents Assigned From Pool

See `agents_from_pool` for information about using this element.

---

## agents\_from\_pool - Agents Assigned From Pool

The number of agents assigned from the agent pool.

*Table 202. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

This element can be used with the `agents_created_empty_pool` monitor element to determine how often an agent must be created because the pool is empty.

The following ratio

Agents Created Due to Empty Agent Pool / Agents Assigned From Pool

can be used to help set an appropriate value for the `num_poolagents` configuration parameter.

For most users, the default value of 100 with AUTOMATIC will ensure optimal performance.

This ratio may fluctuate somewhat with the workload. At times of low activity on the system, additional agent creation and termination may occur. At times of high activity on the system, more agent reuse will occur. A low ratio indicates that there is a high amount of agent reuse, which is expected on systems with high activity. A high ratio indicates a higher amount of agent creation than reuse is occurring. If this is a concern, increase the value for the `num_poolagents` configuration parameter to lower the ratio. However, this will cause additional resources consumption on the system.

---

## agents\_registered - Agents Registered

The number of agents registered in the database manager instance that is being monitored (coordinator agents and subagents).

*Table 203. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

Use this element to help evaluate your settings for the `max_coordagents` and `max_connections` configuration parameters, as well as the intraquery parallelism settings.

---

## agents\_registered\_top - Maximum Number of Agents Registered

The maximum number of agents that the database manager has ever registered, at the same time, since it was started (coordinator agents and subagents).

Table 204. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

You may use this element to help you evaluate your settings for the **max\_coordagents** and **max\_connections** configuration parameters, as well as the intraquery parallelism settings.

The number of agents registered at the time the snapshot was taken is recorded by the **agents\_registered** monitor element.

---

## agents\_stolen - Stolen Agents

At the database manager snapshot level, this monitor element represents the number of idle agents associated with an application which get reassigned to work on a different application. At the application snapshot level, this monitor element represents the number of idle agents associated with a different application which get reassigned to work on this application.

Table 205. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| Application      | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

### Usage

The **num\_poolagents** configuration parameter is set to AUTOMATIC by default. This means that DB2 automatically manages the pooling of idle agents, which includes assigning work to idle agents associated with another application.

---

## agents\_top - Number of Agents Created

At the application level, this is the maximum number of agents that were used when executing the statement. At the database level, it is the maximum number of agents for all applications.

**Element identifier**  
agents\_top

**Element type**  
watermark

Table 206. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |
| Application    | stmt                  | Statement      |

**Usage** An indicator how well intra-query parallelism was realized.

---

## agents\_waiting\_on\_token - Agents Waiting for a Token

The number of agents waiting for a token so they can execute a transaction in the database manager.

**Note:** The **agents\_waiting\_on\_token** monitor element is deprecated starting with DB2 Version 9.5. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

Table 207. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

You can use this element to help evaluate your setting for the **maxcagents** configuration parameter.

Each application has a dedicated coordinator agent to process database requests within the database manager. Each agent has to get a token before it can execute a transaction. The maximum number of agents that can execute database manager transactions is limited by the configuration parameter **maxcagents**.

---

## agents\_waiting\_top - Maximum Number of Agents Waiting monitor element

The maximum number of agents that have ever been waiting for a token, at the same time, since the database manager was started.

**Note:** The **agents\_waiting\_top** monitor element is deprecated starting with DB2 Version 9.5. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

Table 208. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

Use this element to help you evaluate your setting of the **maxcagents** configuration parameter.

The number of agents waiting for a token at the time the snapshot was taken is recorded by the **agents\_waiting\_on\_token** monitor element.

If the **maxcagents** parameter is set to its default value (-1), no agents should wait for a token and the value of this monitor element should be zero.

---

## agg\_temp\_tablespace\_top - Aggregate temporary table space top monitor element

The high watermark in KB for the aggregate temporary table space usage of DML activities at all nesting levels in a service class. The aggregate is computed by summing the temporary table space usage across all activities in the service subclass, and this high watermark represents the highest value reached by this aggregate since the last reset. The monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE. An AGGSQLTEMPSPACE threshold must be defined and enabled for at least one service subclass in the same superclass as the subclass to which this record belongs, otherwise a value of 0 is returned.

*Table 209. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_scstats         | Always collected |

### Usage

Use this element to determine the highest aggregate DML activity system temporary table space usage reached on a member for a service subclass in the time interval collected.

---

## aggsqtempespace\_threshold\_id - Aggregate SQL temporary space threshold ID monitor element

The numeric ID of the AGGSQLTEMPSPACE threshold that was applied to the activity.

*Table 210. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand which AGGSQLTEMPSPACE threshold, if any, was applied to the activity.

---

## aggsqtempespace\_threshold\_value - AggSQL temporary space threshold value monitor element

The upper bound of the AGGSQLEMPSPACE threshold that was applied to the activity.

*Table 211. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand the value of the AGGSQLEMPSPACE threshold applied to the activity, if any.

---

## aggsqtempespace\_threshold\_violated - AggSQL temporary space threshold violated monitor element

The optional monitor element when set to 'Yes' indicates that the activity violated the AGGSQLEMPSPACE threshold that was applied to it. 'No' indicates that the activity has not yet violated the threshold.

*Table 212. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to determine if the activity violated the AGGSQLEMPSPACE threshold that was applied to the activity.

---

## app\_act\_aborted\_total - Total failed external coordinator activities monitor element

The total number of external, non-nested coordinator activities that completed with errors. For service classes, if an activity is remapped to a different service subclass with a REMAP ACTIVITY action before it aborts, the activity counts only toward the total of the subclass in which it aborts.

*Table 213. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics                     | REQUEST METRICS BASE   |

Table 213. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element  | Collection Level |
|--|------------------|------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS  | BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                                    | ACTIVITY METRICS | BASE             |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics   | ACTIVITY METRICS | BASE             |

Table 214. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## app\_act\_completed\_total - Total successful external coordinator activities monitor element

The total number of external, non-nested coordinator activities that completed successfully.

For service classes, if an activity is remapped to a different subclass with a REMAP ACTIVITY action before it completes, this activity counts only toward the total of the subclass it completes in.

Table 215. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                                    | ACTIVITY METRICS BASE  |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics   | ACTIVITY METRICS BASE  |

Table 216. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## app\_act\_rejected\_total - Total rejected external coordinator activities monitor element

The total number of external, non-nested coordinator activities at any nesting level that were rejected instead of being allowed to execute.

Table 217. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                                    | ACTIVITY METRICS BASE  |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics   | ACTIVITY METRICS BASE  |

Table 218. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

---

## appl\_action - Application action monitor element

The action or request that the client application is performing.

Table 219. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## app\_rqsts\_completed\_total - Total application requests completed monitor element

Total number of external (application) requests executed by the coordinator. For service subclasses, this monitor element is updated only for the subclass where the application request completes.

Table 220. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 221. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## Usage

Use this monitor element to understand how many requests are being submitted into the system from applications.

---

### appl\_con\_time - Connection Request Start Timestamp

The date and time that an application started a connection request.

**Element identifier**

appl\_con\_time

**Element type**

timestamp

*Table 222. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Timestamp      |

**Usage** Use this element to determine when the application started its connection request to the database.

---

### appl\_id - Application ID monitor element

This identifier is generated when the application connects to the database at the database manager or when DB2 Connect receives a request to connect to a DRDA database.

*Table 223. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function -<br>Get connection metrics  | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)           | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)    | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return<br>records from a given facility  | ACTIVITY METRICS BASE            |
| PDLOGMSG_LAST24HOURS<br>administrative view and<br>PD_GET_LOG_MSGS table function -<br>Retrieve problem determination messages | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table<br>function - list agents running in a service<br>class                                     | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list<br>workload occurrences                                   | ACTIVITY METRICS BASE            |

Table 224. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| DCS Application | dcs_appl_info         | Basic          |
| Lock            | appl_lock_list        | Basic          |

Table 225. Event Monitoring Information

| Event Type                          | Logical Data Grouping                    | Monitor Switch   |
|-------------------------------------|--|------------------|
| Locking <sup>1</sup>                | lock_participants                        | Always collected |
| Unit of work <sup>1</sup>           | uow, uow_executable_list,<br>uow_metrics | Always collected |
| Connection                          | event_conn                               | Always collected |
| Connections                         | event_connheader                         | Always collected |
| Statements                          | event_stmt                               | Always collected |
| Transactions <sup>2</sup>           | event_xact                               | Always collected |
| Deadlocks <sup>3</sup>              | event_dlconn                             | Always collected |
| Deadlocks with Details <sup>3</sup> | event_detailed_dlconn                    | Always collected |
| Activities                          | event_activitystmt                       | Always collected |
| Activities                          | event_activity                           | Always collected |
| Activities                          | event_activityvals                       | Always collected |
| Activities                          | event_activitymetrics                    | Always collected |
| Threshold violations                | event_thresholdviolations                | Always collected |
| Change history                      | changesummary                            | Always collected |

1. For this event monitor, this monitor element is returned in the column APPLICATION\_ID.
2. This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR UNIT OF WORK statement to monitor transaction events.
3. This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This ID is known on both the client and server, so you can use it to correlate the client and server parts of the application. For DB2 Connect applications, you will also need to use **outbound\_app1\_id** monitor element to correlate the client and server parts of the application.

This identifier is unique across the network. There are different formats for the application ID, which are dependent on the communication protocol between the client and the server machine on which the database manager, DB2 Connect, or both are running. Each of the formats consists of three parts separated by periods.

1. TCP/IP

**Format**

IPAddr.Port.Timestamp

**IPv4****Example**

9.26.120.63.43538.090924175700

**Details**

In IPv4, a TCP/IP-generated application ID is composed of three sections. The first section is the IP address. It is represented as four decimal numbers of the form a.b.c.d. The second section is the port number, which is represented as 5 decimal characters. The third section is the approximate timestamp, represented as 12 decimal characters.

**IPv6****Example**

2002:91a:519:13:20d:60ff:feef:cc64.5309.090924175700

**Details**

In IPv6, a TCP/IP-generated application ID is composed of three sections. The first section contains the IPv6 address of the form a:b:c:d:e:f:g:h, where each of a-h is up to 4 hexadecimal digits. The second section is the port number. The third section is the approximate timestamp identifier for the instance of this application.

**2. Local Applications****Format**

\*LOCAL.DB2 instance.Application instance

**Example**

\*LOCAL.DB2INST1.930131235945

**Details**

The application ID generated for a local application is made up by concatenating the string \*LOCAL, the name of the DB2 instance, and a unique identifier for the instance of this application.

For multiple database partition instances, LOCAL is replaced with Nx, where x is the partition number from which the client connected to the database. For example, \*N2.DB2INST1.0B5A12222841.

Use the **client\_protocol** monitor element to determine which communications protocol the connection is using and, as a result, the format of the **appl\_id** monitor element.

---

## appl\_id\_holding\_lk - Application ID Holding Lock

The application ID of the application that is holding a lock on the object that this application is waiting to obtain.

**Element identifier**

appl\_id\_holding\_lk

**Element type**

information

Table 226. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock_wait             | Lock           |

Table 227. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** This element can help you determine which applications are in contention for resources. Specifically, it can help you identify the application handle (agent ID) and table ID that are holding the lock. Note that you may use the LIST APPLICATIONS command to obtain information to relate the application ID with an agent ID. However, it is a good idea to collect this type of information when you take the snapshot, as it could be unavailable if the application ends before you run the LIST APPLICATIONS command.

Note that more than one application can hold a shared lock on an object for which this application is waiting to obtain a lock. See lock\_mode for information about the type of lock that the application holds. If you are taking an application snapshot, only one of the application IDs holding a lock on the object will be returned. If you are taking a lock snapshot, all of the application IDs holding a lock on the object will be returned.

---

## appl\_id\_oldest\_xact - Application with Oldest Transaction

The application ID (which corresponds to the *agent\_id* value from the application snapshot) of the application that has the oldest transaction.

**Element identifier**

appl\_id\_oldest\_xact

**Element type**

information

Table 228. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** This element can help you determine which application has the oldest active transaction. This application can be forced to free up log space. If it is taking up a great deal of log space, you should examine the application to determine if it can be modified to commit more frequently.

There are times when there is not a transaction holding up logging, or the oldest transaction does not have an application ID (for example, indoubt transaction or inactive transaction). In these cases, this application's ID is not returned in the data stream.

---

## appl\_idle\_time - Application Idle Time

Number of seconds since an application has issued any requests to the server. This includes applications that have not terminated a transaction, for example not issued a commit or rollback.

**Element identifier**

appl\_idle\_time

**Element type**

information

*Table 229. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl                  | Statement      |
| DCS Application | dcs_appl              | Statement      |

**Usage** This information can be used to implement applications that force users that have been idle for a specified number of seconds.

---

## appl\_name - Application name monitor element

The name of the application running at the client, as known to the database or DB2 Connect server.

*Table 230. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS table function - Retrieve column information for temporary tables | ACTIVITY METRICS BASE            |
| ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables          | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics  | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences   | ACTIVITY METRICS BASE            |

*Table 231. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 232. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Locking        | -                     | Always collected |
| Unit of work   | -                     | Always collected |
| Connections    | event_connheader      | Always collected |
| Activities     | event_activity        | Always collected |
| Change history | changesummary         | Always collected |

## Usage

This element can be used with **appl\_id** to relate data items with your application.

In a client-server environment, this name is passed from the client to the server when establishing the database connection. Any non-English characters in the application name will be removed. A CLI application can set the `SQL_ATTR_INFO_PROGRAMNAME` attribute with a call to `SQLSetConnectAttr`. When `SQL_ATTR_INFO_PROGRAMNAME` is set before the connection to the server is established, the value specified overrides the actual client application name and will be the value that is displayed in the **appl\_name** monitor element.

In situations where the client application code page is different from the code page under which the database system monitor is running, you can use **codepage\_id** to help translate **appl\_name**.

---

## appl\_priority - Application Agent Priority

The priority of the agents working for this application.

### Element identifier

appl\_priority

### Element type

information

Table 233. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 234. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** You can use this element to check if applications are running with the expected priorities. Application priorities can be set by an administrator. They can be changed by the governor utility (**db2gov**).

The governor is used by DB2 to monitor and change the behavior of applications running against a database. This information is used to schedule applications and balance system resources.

A governor daemon collects statistics about the applications by taking snapshots. It checks these statistics against the rules governing applications

running on that database. If the governor detects a rule violation, it takes the appropriate action. These rules and actions were specified by you in the governor configuration file.

If the action associated with a rule is to change an application's priority, the governor changes the priority of the agents in the partition where the violation was detected.

---

## appl\_priority\_type - Application Priority Type

Operating system priority type for the agent working on behalf of the application.

### Element identifier

appl\_priority\_type

### Element type

information

*Table 235. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

*Table 236. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** Dynamic priority is recalculated by the operating system based on usage. Static priority does not change.

---

## appl\_section\_inserts - Section Inserts monitor element

Inserts of SQL sections by an application from its shared SQL workspace.

*Table 237. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

*Table 238. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

### Usage

The working copy of any executable section is stored in a shared SQL workspace. This is a count of when a copy was not available and had to be inserted.

---

## appl\_section\_lookups - Section Lookups

Lookups of SQL sections by an application from its shared SQL workspace.

Table 239. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 240. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

### Usage

Each agent has access to a shared SQL workspace where the working copy of any executable section is kept. This counter indicates how many times the SQL work area was accessed by agents for an application.

---

## appl\_status - Application status monitor element

The current status of the application.

Table 241. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_id_info          | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 242. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Locking    | -                     | Always collected |
| Connection | event_conn            | Always collected |

### Usage

This element can help you diagnose potential application problems. Values for this field are listed in the following table.

| API Constant         | Description   |
|----------------------|---|
| SQLM_AUTONOMOUS_WAIT | <b>Autonomous Wait:</b> The application is waiting for an autonomous routine to complete. |
| SQLM_BACKUP          | <b>Backing Up Database:</b> The application is performing a backup of the database.       |
| SQLM_COMMIT_ACT      | <b>Commit Active:</b> The unit of work is committing its database changes.                |

| API Constant            | Description  |
|-------------------------|--|
| SQLM_COMP               | <b>Compiling:</b> The database manager is compiling an SQL statement or precompiling a plan on behalf of the application.  |
| SQLM_CONNECTED          | <b>Database Connect Completed:</b> The application has initiated a database connection and the request has completed.  |
| SQLM_CONNECTPEND        | <b>Database Connect Pending:</b> The application has initiated a database connection but the request has not yet completed.  |
| SQLM_CREATE_DB          | <b>Creating Database:</b> The agent has initiated a request to create a database and that request has not yet completed.   |
| SQLM_DECOUPLED          | <b>Decoupled from Agent:</b> There are no agents currently associated with the application. This is a normal state. When the Connection Concentrator is enabled, there is no dedicated coordinator agent, so an application can be decoupled on the coordinator partition. In non-concentrator environments, an application cannot be decoupled on the coordinator partition as there will always be a dedicated coordinator agent . |
| SQLM_DISCONNECTPEND     | <b>Database Disconnect Pending:</b> The application has initiated a database disconnect but the command has not yet completed executing. The application may not have explicitly executed the database disconnect command. The database manager will disconnect from a database if the application ends without disconnecting.   |
| SQLM_INTR               | <b>Request Interrupted:</b> An interrupt of a request is in progress.  |
| SQLM_IOERROR_WAIT       | <b>Wait to Disable Table space:</b> The application has detected an I/O error and is attempting to disable a particular table space. The application has to wait for all other active transactions on the table space to complete before it can disable the table space.   |
| SQLM_LOAD               | <b>Data Fast Load:</b> The application is performing a "fast load" of data into the database.  |
| SQLM_LOCKWAIT           | <b>Lock Wait:</b> The unit of work is waiting for a lock. After the lock is granted, the status is restored to its previous value.   |
| SQLM QUIESCE_TABLESPACE | <b>Quiescing a Table space:</b> The application is performing a quiesce table space request.   |
| SQLM_RECOMP             | <b>Recompiling:</b> The database manager is recompiling (that is, rebinding) a plan on behalf of the application.  |
| SQLM_REMOTE_RQST        | <b>Federated request pending:</b> The application is waiting for results from a federated data source.   |
| SQLM_RESTART            | <b>Restarting Database:</b> The application is restarting a database in order to perform crash recovery.   |
| SQLM_RESTORE            | <b>Restoring Database:</b> The application is restoring a backup image to the database.  |

| API Constant               | Description  |
|----------------------------|--|
| SQLM_ROLLBACK_ACT          | <b>Rollback Active:</b> The unit of work is rolling back its database changes.   |
| SQLM_ROLLBACK_TO_SAVEPOINT | <b>Rollback to savepoint:</b> The application is rolling back to a savepoint.  |
| SQLM_TEND                  | <b>Transaction Ended:</b> The unit of work is part of a global transaction that has ended but has not yet entered the prepared phase of the two-phase commit protocol.   |
| SQLM_THABRT                | <b>Transaction Heuristically Rolled Back:</b> The unit of work is part of a global transaction that has been heuristically rolled-back.  |
| SQLM_THCOMT                | <b>Transaction Heuristically Committed:</b> The unit of work is part of a global transaction that has been heuristically committed.  |
| SQLM_TPREP                 | <b>Transaction Prepared:</b> The unit of work is part of a global transaction that has entered the prepared phase of the two-phase commit protocol.  |
| SQLM_UNLOAD                | <b>Data Fast Unload:</b> The application is performing a "fast unload" of data from the database.  |
| SQLM_UOWEXEC               | <b>Unit of Work Executing:</b> The database manager is executing requests on behalf of the unit of work.   |
| SQLM_UOWQUEUED             | <b>Unit of Work queued:</b> The unit of work is queued, waiting for another activity to complete execution. The unit of work is queued because the threshold for the number of concurrently executing activities has been reached. |
| SQLM_UOWWAIT               | <b>Unit of Work waiting:</b> The database manager is waiting on behalf of the unit of work in the application. This status typically means that the system is executing in the application's code.                                 |
| SQLM_WAITFOR_REMOTE        | <b>Pending remote request:</b> The application is waiting for a response from a remote partition in a partitioned database instance.   |

## application\_handle - Application handle monitor element

A system-wide unique ID for the application. In a single-member database configuration, this identifier consists of a 16-bit counter. In a multi-member configuration, this identifier consists of the coordinating member number concatenated with a 16-bit counter. In addition, this identifier is the same on every member where the application may make a secondary connection.

Table 243. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |

Table 243. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database                                  | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 244. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcs_appl_info         | Basic          |
| Transaction     | event_xact            | -              |

Table 245. Event Monitoring Information

| Event Type                          | Logical Data Grouping     | Monitor Element Collection Level |
|-------------------------------------|---------------------------|----------------------------------|
| Locking                             | -                         | Always collected                 |
| Unit of work                        | -                         | Always collected                 |
| Connections                         | event_connheader          | Always collected                 |
| Statements                          | event_stmt                | Always collected                 |
| Statements                          | event_subsection          | Always collected                 |
| Deadlocks <sup>1</sup>              | event_dlconn              | Always collected                 |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn     | Always collected                 |
| Threshold violations                | event_thresholdviolations | Always collected                 |
| Activities                          | event_activity            | Always collected                 |
| Change history                      | changesummary             | Always collected                 |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This monitor element is an alias of the **agent\_id** monitor element.

When returned by MON\_GET\_MEMORY\_POOL, this monitor element is NULL except when the memory pool being described is one of the following types:

- APPLICATION
- STATISTICS
- STATEMENT
- SORT\_PRIVATE.

---

## appls\_cur\_cons - Applications Connected Currently

Indicates the number of applications that are currently connected to the database.

Table 246. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Lock           | db_lock_list          | Basic          |

**Usage** You may use this element to help you understand the level of activity within a database and the amount of system resource being used.

It can help you adjust the setting of the *maxappls* and *max\_coordagents* configuration parameters. For example, its value is always the same as *maxappls*, you may want to increase the value of *maxappls*. See the *rem\_cons\_in* and the *local\_cons* monitor elements for more information.

---

## appls\_in\_db2 - Applications Executing in the Database Currently

Indicates the number of applications that are currently connected to the database, and for which the database manager is currently processing a request.

Table 247. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

---

## arm\_correlator - Application response measurement correlator monitor element

Identifier of a transaction in the Application Response Measurement (ARM) standard.

Table 248. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

### Usage

This element can be used to link an activity collected by the activities event monitor to the applications associated with the activity, if such applications also support the Application Response Measurement (ARM) standard.

---

## associated\_agents\_top - Maximum Number of Associated Agents

The maximum number of subagents associated with this application.

Table 249. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

---

## async\_read\_time - Asynchronous read time monitor element

The total amount of time that asynchronous engine dispatchable units (EDUs) spent reading from the buffer pool or table space. This value is reported in milliseconds.

Table 250. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

---

## async\_write\_time - Asynchronous write time monitor element

The total amount of time that asynchronous engine dispatchable units (EDUs) spent writing to the buffer pool or table space. This value is reported in milliseconds.

Table 251. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

---

## async\_runstats – Total number of asynchronous RUNSTATS requests monitor element

The total number of successful asynchronous RUNSTATS activities performed by real-time statistics gathering for all the applications in the database. Values reported by all the database partitions are aggregated together.

Table 252. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

Table 253. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

Table 254. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

## Usage

Use this element to determine how many successful asynchronous RUNSTATS activities have been performed by real-time statistics gathering. This value changes frequently. In order to get a better view of the system usage, take a snapshot at specific intervals over an extended period of time. When used in conjunction with **sync\_runstats** and **stats\_fabrications** monitor elements, this element can help you to track the different types of statistics collection activities related to real-time statistics gathering and analyze their performance impact.

---

## audit\_events\_total - Total audit events monitor element

The total number of audit events generated.

Table 255. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 255. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 256. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## audit\_file\_write\_wait\_time - Audit file write wait time monitor element

Time spent waiting to write an audit record. The value is given in milliseconds.

Table 257. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 257. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 258. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the amount of time an agent spends waiting to open and write an audit event synchronously to disk.

In a typical scenario, only one agent attempts to open the audit log file at a time, as the other agents wait for access to the audit common subsystem before opening the file. Therefore, the wait time usually represents the time spent waiting to write the file to disk by the operating system. Audit utilities might lock the audit log file during execution, which causes a longer than normal wait time for agents to open and write to the audit log file. If asynchronous auditing is enabled, audit events

that are larger than the asynchronous audit buffer are written directly to disk, instead of to the buffer, and contribute to the wait time.

Outside of the special audit utility scenario, the wait time depends on the speed of the disks and how quickly the operating system can write the data to them. In order to reduce this wait time for a given application and audit configuration, you might tune the operating system or use faster disks.

---

## audit\_file\_writes\_total - Total audit files written monitor element

The total number of times an agent has had to wait to write an audit event directly to disk.

*Table 259. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 260. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element in conjunction with the `audit_file_write_wait_time` monitor element to determine the average time an application request spends waiting to open and write an audit event synchronously to disk.

## audit\_subsystem\_wait\_time - Audit subsystem wait time monitor element

Time spent waiting for space in audit buffer. Waiting occurs when audit buffer is full and agent must wait for audit daemon to write buffer to disk. The value is given in milliseconds.

Table 261. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 261. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element  | Collection Level |
|---|------------------|------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS | BASE             |
| MON_GET_SERVICE_SUBCLASS<br>table function - Get service subclass metrics   | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK<br>table function - Get unit of work metrics   | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD<br>table function - Get workload metrics   | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS<br>table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS  | BASE             |

Table 262. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the amount of time an agent spends waiting to access the audit common subsystem, while the audit common subsystem is busy handling events for other agents.

Certain common portions of the audit subsystem can only be accessed by a single agent at a time. The value of this monitor element indicates the amount of time that an agent must wait to access the audit common subsystem. This includes time spent by an agent that has filled the current asynchronous buffer waiting for the audit daemon to finish writing out a previous asynchronous buffer to disk. Other agents that are waiting while writing to the audit log file or waiting to make a request of the audit daemon have also accessed the audit common subsystem and wait times there will be reflected in this value.

To reduce this wait time, you might change the value of the `audit_buf_sz` configuration parameter if asynchronous auditing is in use. You can increase the value of the `audit_buf_sz` configuration parameter until further increases no longer show any reductions in the audit common subsystem wait time. At this point, the asynchronous buffers are large enough such that the daemon is able to write one full buffer to disk before the next buffer is full, and then the daemon is no longer a bottleneck. If the value of the `audit_buf_sz` configuration parameter must be increased to such an extent that too many audit records could be lost if a system failure were to occur, then you might reduce the wait time by tuning the operating system or using faster disks. If further reduction in the wait time is necessary, then use audit policies to reduce the number of audit events generated.

---

## audit\_subsystem\_waits\_total - Total audit subsystem waits monitor element

Number of times audit has waited for a buffer write.

*Table 263. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 264. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the total number of times an agent has had to access the audit common subsystem. The generation of one audit event may need to access the audit common subsystem none, one, or more times to record the event. Use the **audit\_events\_total** monitor element to determine the exact number of audit events generated.

## auth\_id - Authorization ID

The authorization ID of the user who invoked the application that is being monitored. On a DB2 Connect gateway node, this is the user's authorization ID on the host.

Table 265. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| PD_GET_DIAG_HIST table function - Return records from a given facility | ACTIVITY METRICS BASE            |

Table 266. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 267. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Locking      | -                     | Always collected |
| Unit of work | -                     | Always collected |
| Connections  | event_connheader      | Always collected |

## Usage

In an explicit trusted connection, the `auth_id` value does not change immediately when you switch users. Rather, the `auth_id` is updated the first time you access the database after switching users. This is because the switch user operation is always chained to the subsequent operation.

You can use this element to determine who invoked the application.

---

## authority\_bitmap - User authorization level monitor element

The authorities granted to the user and to the groups to which the user belongs. These include authorities granted to roles that are granted to the user and to the groups to which the user belongs. Authorities granted to a user or to roles granted to the user are considered user authorities. Authorities granted to a group to which the user belongs or to roles granted to the group to which the user belongs are considered group authorities.

Table 268. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | appl_info             | Basic          |

Table 269. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

## Usage

The `authority_bitmap` monitor element has the format of an array. Each array element is a single character that represents whether or not the user ID has been granted a specific authority and how the user has received that authority.

Individual array elements are indexed through an index value defined in the `sql.h` file. The value of an index in the `authority_bitmap` array is called an *authority index*. For example, `SQL_DBAUTH_SYSADM` is the index to determine if the user has `SYSADM` authority.

The value of one element in the `authority_bitmap` array identified by an authority index represents whether the authority is held by an authorization ID. To determine how the authorization ID is held, for each array element identified by the authority index, use the following defines from `sql.h`:

### **SQL\_AUTH\_ORIGIN\_USER**

If this bit is on, then the authorization ID has the authority granted to the user or to a role granted to the user.

### **SQL\_AUTH\_ORIGIN\_GROUP**

If this bit is on, then the authorization ID has the authority granted to the group or to a role granted to the group.

For example, to determine if a user holds `DBADM` authority, verify the following value:

```
authority_bitmap[SQL_DBAUTH_DBADM]
```

To determine if the DBADM authority is held directly by the user, verify the following:

```
authority_bitmap[SQL_DBAUTH_DBADM] & SQL_AUTH_ORIGIN_USER
```

---

## authority\_lvl - User authorization level monitor element

The highest authority level granted to an application.

**Note:** The authority\_lvl monitor element is deprecated starting with DB2 database Version 9.5. Use the authority\_bitmap monitor element instead. See “authority\_bitmap - User authorization level monitor element” on page 687.

Table 270. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | appl_info             | Basic          |

Table 271. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** The operations allowed by an application are granted either directly or indirectly.

The following defines from sql.h may be used to determine the authorizations granted explicitly to a user:

- SQL\_SYSADM
- SQL\_DBADM
- SQL\_CREATETAB
- SQL\_BINDADD
- SQL\_CONNECT
- SQL\_CREATE\_EXT\_RT
- SQL\_CREATE\_NOT\_FENC
- SQL\_SYSCTRL
- SQL\_SYSMANT

The following defines from sql.h may be used to determine indirect authorizations inherited from group or public:

- SQL\_SYSADM\_GRP
- SQL\_DBADM\_GRP
- SQL\_CREATETAB\_GRP
- SQL\_BINDADD\_GRP
- SQL\_CONNECT\_GRP
- SQL\_CREATE\_EXT\_RT\_GRP
- SQL\_CREATE\_NOT\_FENC\_GRP
- SQL\_SYSCTRL\_GRP
- SQL\_SYSMANT\_GRP

---

## auto\_storage\_hybrid - Hybrid automatic storage table space indicator monitor element

If the table space is an automatic storage table space with some non-automatic storage containers, this monitor element returns a value of 1. Otherwise, it returns a value of 0.

Table 272. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

### Usage

A hybrid automatic storage table space is a table space that has been converted to be managed by automatic storage using the ALTER TABLESPACE command, but has not yet been rebalanced. This table space still has non-automatic storage containers. After the table space is rebalanced, it contains only automatic storage containers, and is no longer considered a hybrid table space.

---

## automatic - Buffer pool automatic monitor element

Indicates whether a particular buffer pool has self-tuning enabled. This element is set to 1 if self-tuning is enabled for the buffer pool; and 0 otherwise.

Table 273. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE                     |

---

## backup\_timestamp - Backup timestamp

Timestamp of the backup image.

Table 274. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Change History | changesummary         | Always collected                 |

### Usage

For the change history event monitor:

- If UTILITY\_TYPE is BACKUP and EVENT\_TYPE is UTILSTART, the BACKUP\_TIMESTAMP value is the timestamp of the backup image. If UTILITY\_TYPE is RESTORE and EVENT\_TYPE is UTILSTOP, the BACKUP\_TIMESTAMP value is the timestamp of the backup image. For all other cases, the BACKUP\_TIMESTAMP is an empty string.
- For RESTORE, the image timestamp is not always known at the start time of the utility.

A BACKUP\_TIMESTAMP can be correlated with information stored in the database history file (for example, Lookup sequence information) using the SYSIBMADM.DB\_HISTORY administration view

---

## bin\_id - Histogram bin identifier monitor element

The identifier of a histogram bin. The **bin\_id** is unique within a histogram.

Table 275. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

### Usage

Use this element to distinguish bins within the same histogram.

---

## binds\_precompiles - Binds/Precompiles Attempted

The number of binds and pre-compiles attempted.

#### Element identifier

binds\_precompiles

#### Element type

counter

Table 276. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 277. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to gain insight into the current level of activity within the database manager.

This value does not include the count of *int\_auto\_rebinds*, but it does include binds that occur as a result of the REBIND PACKAGE command.

---

## block\_ios - Number of block I/O requests monitor element

The number of block I/O requests. More specifically, the number of times DB2 performs sequential prefetching of pages into the block area of the buffer pool.

Table 278. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |

Table 278. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONTAINER table function -<br>Get table space container metrics | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics          | DATA OBJECT METRICS BASE         |

Table 279. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

## Usage

If block-based buffer pool is enabled, this monitor element will report how often block I/O is being done. Otherwise, this monitor element will return 0. The number of block I/O requests is monitored only during sequential prefetching when using block-based buffer pools.

If block-based buffer pool is enabled and this number is very low, or close to the number of vectored I/Os (the value of the **vectored\_ios** monitor element), consider changing the block size. This state can be an indication of the following occurrences:

- The extent size of one or more table spaces bound to the buffer pool is smaller than the block size specified for the buffer pool.
- Some pages requested in the prefetch request are already present in the page area of the buffer pool.

The prefetcher allows some wasted pages in each buffer pool block, but if too many pages are wasted, then the prefetcher will decide to perform vectored I/O into the page area of the buffer pool.

To take full advantage of the sequential prefetch performance improvements that block-based buffer pools provide, it is essential to choose an appropriate value for the block size. This can, however, be difficult because multiple table spaces with different extent sizes can be bound to the same block-based buffer pool. For optimal performance, it is recommended that you bind table spaces with the same extent size to a block-based buffer pool with a block size equal to the extent size. Good performance can be achieved when the extent size of the table spaces are greater than the block size, but not when the extent size is smaller than the block size.

For example, if the extent size is 2 and the block size is 8, vectored I/O would be used instead of block I/O (block I/O would have wasted 6 pages). A reduction of the block size to 2 would solve this problem.

---

## blocking\_cursor - Blocking Cursor

This element indicates if the statement being executed is using a blocking cursor.

### Element identifier

blocking\_cursor

**Element type**  
information

*Table 280. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

*Table 281. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks with Details | event_detailed_dlconn | Always collected |
| Statements             | event_stmt            | Always collected |

**Usage** Using blocking for data transfer for a query can improve its performance. The SQL used for a query can affect the use of blocking and might require some modification.

---

## **blocks\_pending\_cleanup - Pending cleanup rolled-out blocks monitor element**

The total number of MDC table blocks in the database that are pending asynchronous cleanup following a roll out delete.

*Table 282. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information | ACTIVITY METRICS BASE            |

*Table 283. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |
| Database       | event_db              | -              |

### **Usage**

Use this element to determine the number of MDC table blocks that, following the deletion of a defer cleanup roll out, have not been released back to the system as available storage.

---

## **bottom - Histogram bin bottom monitor element**

The exclusive bottom end of the range of a histogram bin. The value of this monitor element is also the top inclusive end of the range of the previous histogram bin, if there is one.

*Table 284. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

## Usage

Use this element with the corresponding **top** element to determine the range of a bin within a histogram.

---

### boundary\_leaf\_node\_splits - Boundary leaf node splits monitor element

Number of times a boundary leaf node was split during an insert operation.

*Table 285. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

### bp\_cur\_buffsz - Current Size of Buffer Pool

The current buffer pool size, in pages.

*Table 286. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | ACTIVITY METRICS BASE            |

*Table 287. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

---

### bp\_id - Buffer pool identifier monitor element

This element contains the buffer pool identifier for the buffer pool that is being monitored.

*Table 288. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Basic          |

---

### bp\_name - Buffer pool name monitor element

The name of the buffer pool.

*Table 289. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | ACTIVITY METRICS BASE                        |

Table 290. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Basic          |

**Usage** Each database requires at least one buffer pool. Depending on your needs, you may choose to create several buffer pools, each of a different size, for a single database. The CREATE, ALTER, and DROP BUFFERPOOL statements allow you to create, change, or remove a buffer pool.

When a database is created, it has a default buffer pool called IBMDEFAULTBP with a size determined by the platform. It also has a set of system buffer pools, each corresponding to a different page size:

- IBMSYSTEMBP4K
- IBMSYSTEMBP8K
- IBMSYSTEMBP16K
- IBMSYSTEMBP32K

These system buffer pools cannot be altered.

---

## bp\_new\_buffsz - New Buffer Pool Size

The size the buffer pool will be changed to once the database is restarted. When the ALTER BUFFERPOOL statement is executed as DEFERRED, the buffer pool size is not changed until the database is stopped and restarted.

Table 291. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

---

## bp\_pages\_left\_to\_remove - Number of Pages Left to Remove

The number of pages left to remove from the buffer pool before the buffer pool resize is completed. This applies only to buffer pool resize operations invoked by ALTER BUFFERPOOL statements executed as IMMEDIATE.

Table 292. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

---

## bp\_tbsp\_use\_count - Number of Table Spaces Mapped to Buffer Pool

The number of table spaces using this buffer pool.

Table 293. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

---

## buff\_auto\_tuning - FCM buffer auto-tuning indicator monitor element

Indicates whether the number of fast communication manager (FCM) buffers is set and tuned automatically. A value of 1 means "Yes"; a value of 0 means "No".

Table 294. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

### Usage

FCM buffer auto-tuning is enabled by setting the **fcm\_num\_buffers** configuration parameter to AUTOMATIC.

---

## buff\_free - FCM Buffers Currently Free

This element indicates the number of FCM buffers currently free.

### Element identifier

buff\_free

### Element type

gauge

Table 295. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 296. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

### Usage

To calculate the percentage of free FCM buffers, use the following formula:  
 $(\text{buff\_free}/\text{buff\_total}) * 100$

If the percentage of free FCM buffers falls below 20% and if the FCM buffer auto-tuning is enabled, then DB2 database manager will adjust the FCM buffer numbers.

If the percentage of free FCM buffers falls below 20% and if the FCM buffer auto-tuning is not enabled, then you need to tune the **fcm\_num\_buffers** configuration parameter.

---

## buff\_free\_bottom - Minimum FCM Buffers Free

The lowest number of free FCM buffers reached during processing.

Table 297. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 298. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

Use this element along with the **fcm\_num\_buffers** configuration parameter to determine the maximum FCM buffer pool utilization. If the value of the **buff\_free\_bottom** monitor element is low, increase the value of the **fcm\_num\_buffers** configuration parameter to ensure that operations do not run out of FCM buffers. If the value of the **buff\_free\_bottom** monitor element is high, decrease the value of the **fcm\_num\_buffers** configuration parameter to conserve system resources.

## buff\_max - Maximum possible number of FCM buffers monitor element

Maximum number of fast communication manager (FCM) buffers that can be allocated, based on the amount of virtual memory reserved when the instance was started.

Table 299. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 300. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

This internal monitor element is used by IBM Support only.

## buff\_total - Number of currently allocated FCM buffers monitor element

Number of fast communication manager (FCM) buffers currently allocated. This number includes both in-use buffers and free buffers.

Table 301. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 302. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

If the **buff\_auto\_tuning** monitor element indicates that FCM is being tuned automatically, then the value of the **buff\_total** monitor element is adjusted based on the demand for FCM buffers.

To determine the number of FCM buffers currently in use, use the following formula:

$\text{buff\_total} - \text{buff\_free}$

To calculate the percentage of free FCM buffers, use the following formula:

$(\text{buff\_free}/\text{buff\_total}) * 100$

If the percentage of free FCM buffers falls below 20% and if the FCM buffer auto-tuning is enabled, then DB2 database manager will adjust the FCM buffer numbers.

If the percentage of free FCM buffers falls below 20% and if the FCM buffer auto-tuning is not enabled, then you need to tune the **fcm\_num\_buffers** configuration parameter.

---

## byte\_order - Byte Order of Event Data

The byte ordering of numeric data, specifically whether the event data stream was generated on a “big endian” server (for example, a RS/6000®) or “little endian” server (for example, an Intel-based PC running Windows 2000).

*Table 303. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |

**Usage** This information is needed to allow you to interpret numeric data in the data stream, since the byte order of integers on a “big endian” server is the reverse of the byte order on a “little endian” server.

If the application that processes the data recognizes that it is running on one type of computer hardware (for example, a big endian computer), while the event data was produced on the other type of computer hardware (for example, a little endian computer), then the monitoring application will have to reverse the bytes of numeric data fields before interpreting them. Otherwise, byte reordering is not required.

This element can be set to one of the following API constants:

- SQLM\_BIG\_ENDIAN
- SQLM\_LITTLE\_ENDIAN

---

## cached\_timestamp - Cached timestamp monitor element

The time when the server list was cached.

*Table 304. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVERLIST table function - get member priority details | Always collected                 |

## cat\_cache\_inserts - Catalog cache inserts monitor element

The number of times that the system tried to insert table descriptor or authorization information into the catalog cache.

*Table 305. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 306. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 307. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Database     | event_db  | Always collected     |
| Connection   | event_conn  | Always collected     |
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## Usage

In conjunction with "Catalog Cache Lookups", you can calculate the catalog cache hit ratio using the following formula:

$$1 - (\text{Catalog Cache Inserts} / \text{Catalog Cache Lookups})$$

See the **cat\_cache\_lookups** monitor element for more information about using this element.

---

## cat\_cache\_lookups - Catalog cache lookups monitor element

The number of times that the catalog cache was referenced to obtain table descriptor information or authorization information.

*Table 308. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 309. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 310. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Database     | event_db  | Always collected     |
| Connection   | event_conn  | Always collected     |
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## Usage

This element includes both successful and unsuccessful accesses to the catalog cache. The catalog cache is referenced whenever:

- a table, view, or alias name is processed during the compilation of an SQL statement
- database authorization information is accessed
- a routine is processed during the compilation of an SQL statement

To calculate the catalog cache hit ratio use the following formula:

$$(1 - (\text{cat\_cache\_inserts} / \text{cat\_cache\_lookups}))$$

indicates how well the catalog cache is avoiding catalog accesses. If the ratio is high (more than 0.8), then the cache is performing well. A smaller ratio might suggest that the **catalogcache\_sz** configuration parameter should be increased. You should expect a large ratio immediately following the first connection to the database.

The execution of Data Definition Language (DDL) SQL statements involving a table, view, or alias will evict the table descriptor information for that object from the catalog cache causing it to be re-inserted on the next reference. In addition, GRANT and REVOKE statements for database authorization and execute privilege of routines will evict the subject authorization information from the catalog cache. Therefore, the heavy use of DDL statements and GRANT/REVOKE statements may also increase the ratio.

---

## cat\_cache\_overflows - Catalog Cache Overflows

The number of times that the catalog cache overflowed the bounds of its allocated memory.

Table 311. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 312. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

## Usage

Use this element with the **cat\_cache\_size\_top** monitor element to determine whether the size of the catalog cache needs to be increased to avoid overflowing.

Catalog cache space is reclaimed by evicting table descriptor information for tables, views, or aliases, or authorization information that is not currently in use by any transaction.

If the value of the **cat\_cache\_overflows** monitor element is large, the catalog cache may be too small for the workload. Enlarging the catalog cache may improve its performance. If the workload includes transactions which compile a large number of SQL statements referencing many tables, views, aliases, user-defined functions, or stored procedures in a single unit of work, then compiling fewer SQL statements in a single transaction may improve the performance of the catalog cache. Or if the workload includes binding of packages containing many SQL statements referencing many tables, views, aliases, user-defined functions, or stored procedures, you can try splitting packages so that they include fewer SQL statements to improve performance.

---

## cat\_cache\_size\_top - Catalog cache high watermark monitor element

The largest logical size reached by the catalog cache.

Table 313. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 314. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

## Usage

This element indicates the maximum number of bytes the catalog cache required logically for the workload run against the database since it was activated.

The catalog cache is managed by logical size, which does not include memory management usage. The **pool\_watermark** element in the database snapshot provides the physical high water mark value for memory used by the catalog cache. The logical size rather than physical size should be used for catalog cache monitoring and tuning efforts.

If the catalog cache overflowed, then this element contains the largest size reached by the catalog cache during the overflow. Check the **cat\_cache\_overflows** monitor element to determine if such a condition occurred.

You can determine the minimum size of the catalog cache required by your workload by:

$$\text{maximum catalog cache size} / 4096$$

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the catalog cache to avoid overflow.

---

## catalog\_node - Catalog Node Number

The node number of the node where the database catalog tables are stored.

**Element identifier**

catalog\_node

**Element type**

information

*Table 315. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 316. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** The catalog node is the node where all system catalog tables are stored. All access to system catalog tables must go through this node.

---

## catalog\_node\_name - Catalog Node Network Name

The network name of the catalog node.

**Element identifier**

catalog\_node\_name

**Element type**

information

*Table 317. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 318. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element to determine the location of a database.

---

## cf\_waits - Number of cluster caching facility waits monitor element

The number of times that the DB2 database system waited while it communicated with a cluster caching facility.

Table 319. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics                                      | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |

Table 320. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

---

## cf\_wait\_time - cluster caching facility wait time monitor element

The amount of time spent communicating with the cluster caching facility. This time does not include time spent performing any of the processing that may have been requested by, or that may occur as a result of the communications, such as granting locks or performing page reclaims. The unit of measurement is milliseconds.

Table 321. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 321. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics                                      | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |

## Usage

This value is an indicator of the amount of time DB2 has spent waiting while communicating with the cluster caching facility.

---

## cfg\_collection\_type - Configuration collection type

Indicates when the configuration parameter value was collected:

Table 322. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

## Usage

The change history event monitor collected this value as:

- I The initial value that was captured when the event monitor was activated.
- U Updated value

---

## cfg\_name - Configuration name

Name of the configuration parameter.

Table 323. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

## Usage

For the change history event monitor, this element identifies the configuration parameter that was updated as part of a DBCFG or DBMCFG event, or captured at event monitor startup as part of a DBCFGVALUES or DBMCFGVALUES event. These events represent the following occurrences:

## DBCFCG

Changing a database configuration parameter

## DBMCFG

Changing a database manager configuration parameter

## DBCFCGVALUES

Capturing database configuration parameter values at event monitor startup, if a database configuration parameter was changed while the event monitor was inactive

## DBMCFGVALUES

Capturing database manager configuration parameter values at event monitor startup, if a database manager configuration parameter was changed while the event monitor was inactive

---

## cfg\_old\_value - Configuration old value

The old value for the configuration parameter or the in memory configuration parameter value.

*Table 324. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

### Usage

For the change history event monitor:

- If the event is a change to a database configuration parameter (DBCFCG) or a database manager configuration parameter (DBMCFG), this is the old configuration parameter value.
- If the event is a capture of database configuration parameter values (DBCFCGVALUES) or database manager configuration parameter values (DBMCFGVALUES) that changed while the event monitor was inactive, this is the current in-memory configuration parameter value. This is the configuration parameter value currently in use.

---

## cfg\_old\_value\_flags - Configuration old value flags

This flag indicates how the old configuration parameter value was determined.

*Table 325. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

### Usage

For the change history event monitor, this element indicates how the old configuration parameter value was determined:

- AUTOMATIC
- COMPUTED
- NONE

If the event is a capture of database configuration parameter values (DBCFCGVALUES) or database manager configuration parameter values (DBMCFGVALUES) that changed while the event monitor was inactive, the flags represent the current in-memory value for the configuration parameter.

---

## cfg\_value - Configuration value

The new value for the configuration parameter or the on-disk configuration parameter value.

Table 326. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

### Usage

For the change history event monitor:

- If the event is a change to a database configuration parameter (DBCFCG) or a database manager configuration parameter (DBMCFG), this is the new value for the configuration parameter.
- If the event is a capture of database configuration parameter values (DBCFCGVALUES) or database manager configuration parameter values (DBMCFGVALUES) that changed while the event monitor was inactive, this is the on-disk configuration parameter value. The on-disk configuration parameter value is the most current value and might not be in effect yet.

---

## cfg\_value\_flags - Configuration value flags

This flag indicates how the new configuration parameter value was determined.

Table 327. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

### Usage

For the change history event monitor, this element indicates how the new configuration parameter value was determined:

- AUTOMATIC
- COMPUTED
- NONE

If the event is a capture of database configuration parameter values (DBCFCGVALUES) or database manager configuration parameter values (DBMCFGVALUES) that changed while the event monitor was inactive, the flags represent the current on-disk value for the configuration parameter.

---

## ch\_auto\_tuning - FCM channel auto-tuning indicator monitor element

Indicates whether the number of fast communication manager (FCM) channels is set and tuned automatically. A value of 1 means "Yes"; a value of 0 means "No".

Table 328. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

### Usage

FCM channel auto-tuning is enabled by setting the **fcm\_num\_channels** configuration parameter to AUTOMATIC.

---

## ch\_free - Channels Currently Free

This element indicates the number of FCM communication channels that are currently free.

Table 329. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 330. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

### Usage

To calculate the percentage of free FCM channels, use the following formula:

$$(ch\_free/ch\_total) * 100$$

If the percentage of free FCM channels falls below 20% and if the FCM channel auto-tuning is enabled, then DB2 database manager will adjust the FCM channel numbers.

If the percentage of free FCM channels falls below 20% and if the FCM channel auto-tuning is not enabled, then you need to tune the **fcm\_num\_channels** configuration parameter.

---

## ch\_free\_bottom - Minimum Channels Free

The lowest number of free FCM communication channels reached during processing.

Table 331. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 332. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

Use this monitor element in conjunction with the **fcm\_num\_channels** configuration parameter to determine the maximum connection entry utilization.

---

## ch\_max - Maximum possible number of FCM channels monitor element

Maximum number of fast communication manager (FCM) channels that can be allocated based on the amount of virtual memory reserved when the instance was started.

Table 333. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 334. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

This internal monitor element is used by IBM Support only.

---

## ch\_total - Number of currently allocated FCM channels monitor element

Number of fast communication manager (FCM) channels currently allocated. This number includes both in-use channels and free channels.

Table 335. Table Function Monitoring Information

| Table Function                | Monitor Element Collection Level |
|-------------------------------|----------------------------------|
| MON_GET_FCM - Get FCM metrics | ACTIVITY METRICS BASE            |

Table 336. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

## Usage

If the **ch\_auto\_tuning** monitor element indicates that FCM is being tuned automatically, then the value of the **ch\_total** monitor element is adjusted based on the demand for FCM channels.

To determine the number of FCM channels currently in use, use the following formula:

$$\text{ch\_total} - \text{ch\_free}$$

To calculate the percentage of free FCM channels, use the following formula:

$$(\text{ch\_free}/\text{ch\_total}) * 100$$

If the percentage of free FCM channels falls below 20% and if the FCM channel auto-tuning is enabled, then DB2 database manager will adjust the FCM channel numbers.

If the percentage of free FCM channels falls below 20% and if the FCM channel auto-tuning is not enabled, then you need to tune the **fcm\_num\_channels** configuration parameter.

---

## client\_acctng - Client accounting string monitor element

The data passed to the target database for logging and diagnostic purposes, if the `sqlseti` API was issued in this connection. The current value of the `CLIENT_ACCTNG` special register for this connection, unit of work, or activity.

**Note:** This element is reported for the coordinating member only. On remote members, the value reported is a string with a length of 0.

This monitor element is synonymous to the **tpmon\_acc\_str** monitor element. The **client\_acctng** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_acc\_str** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

*Table 337. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                              | ACTIVITY METRICS BASE            |

Table 338. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Locking        | -                     | -                                |
| Unit of work   | -                     | -                                |
| Change history | changesummary         | Always collected                 |

## Usage

Use this element for problem determination and accounting purposes.

---

## client\_applname - Client application name monitor element

Identifies the server transaction program performing the transaction, if the sqleset API was issued in this connection. The current value of the CLIENT\_APPLNAME special register for this connection, unit of work, or activity.

**Note:** This element is reported for the coordinating member only. On remote members, the value reported is a string with a length of 0.

This monitor element is synonymous to the **tpmon\_client\_app** monitor element. The **client\_applname** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_app** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

Table 339. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                              | ACTIVITY METRICS BASE            |

Table 340. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Element Collection Level |
|--------------|-----------------------|----------------------------------|
| Locking      | -                     | -                                |
| Unit of work | -                     | -                                |

Table 340. Event Monitoring Information (continued)

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Change history | changesummary         | Always collected                 |

## Usage

Use this element for problem determination and accounting purposes.

---

## client\_db\_alias - Database Alias Used by Application

The alias of the database provided by the application to connect to the database.

### Element identifier

client\_db\_alias

### Element type

information

Table 341. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_id_info          | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 342. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Connections | event_connheader      | Always collected |

**Usage** This element can be used to identify the actual database that the application is accessing. The mapping between this name and *db\_name* could be done by using the database directories at the client node and the database manager server node.

This is the alias defined within the database manager where the database connection request originated.

This element can also be used to help you determine the authentication type, since different database aliases can have different authentication types.

---

## client\_hostname - Client hostname monitor element

The hostname of the machine the client application is connecting from.

Table 343. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |

Table 343. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 344. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Unit of work   | -                     | Always collected                 |
| Change history | changesummary         | Always collected                 |

## client\_idle\_wait\_time - Client idle wait time monitor element

This monitor element records time spent waiting for the client to send its next request. The value is given in milliseconds.

Table 345. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 346. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | Always collected     |

## Usage

Use this monitor element to determine the amount of time spent waiting for requests from a client, as opposed to working on requests. A high client idle time may indicate performance issues that need to be addressed on the client rather than the server.

---

## client\_nname - Client name monitor element

This monitor element is deprecated. The value returned is not a valid value.

Table 347. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Transaction | event_connheader      |                |
| Statement   | event_connheader      |                |
| Deadlock    | event_connheader      |                |
| Connections | event_connheader      |                |

---

## client\_pid - Client process ID monitor element

The process ID of the client application that made the connection to the database.

Table 348. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                           | ACTIVITY METRICS BASE            |

Table 349. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 350. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Unit of work   | -                     | Always collected                 |
| Connections    | event_connheader      | Always collected                 |
| Change history | changesummary         | Always collected                 |

## Usage

You can use this element to correlate monitor information such as CPU and I/O time to your client application.

In the case of a DRDA AS connection, this element will be set to 0.

---

## client\_platform - Client operating platform monitor element

The operating system on which the client application is running.

Table 351. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 352. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 353. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Unit of work   | -                     | Always collected                 |
| Connections    | event_connheader      | Always collected                 |
| Change history | changesummary         | Always collected                 |

## Usage

This element can be used for problem determination for remote applications. Values for this field can be found in the header file `sqlmon.h`.

---

## client\_port\_number - Client port number monitor element

For TCP/IP connections, the port number on the client machine the application is using to communicate with the database server.

*Table 354. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 355. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Unit of work   | -                     | Always collected                 |
| Change history | changesummary         | Always collected                 |

---

## client\_prdid - Client product and version ID monitor element

The product and version that is running on the client. This monitor element is a synonym for the client\_product\_id monitor element.

*Table 356. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 357. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

*Table 358. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |
| Connections  | event_connheader      | Always collected |

## Usage

You can use this element to identify the product and code version of the IBM data server client. It is in the form PPPVRRM, where:

- PPP identifies the product, which is “SQL” for the DB2 products
- VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)
- RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)
- M identifies a 1-character modification level (0-9 or A-Z).

---

## client\_protocol - Client communication protocol monitor element

The communication protocol that the client application is using to communicate with the server.

*Table 359. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 360. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

*Table 361. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Unit of work   | -                     | Always collected                 |
| Connections    | event_connheader      | Always collected                 |
| Change history | changesummary         | Always collected                 |

## Usage

This element can be used for problem determination for remote applications. Values for this field are:

### SQLM\_PROT\_UNKNOWN

The client is communicating using an unknown protocol. This value will only be returned if future clients connect with an earlier level of the server.

### SQLM\_PROT\_LOCAL

The client is running on the same node as the server and no communications protocol is in use.

### SQLM\_PROT\_TCPIP

TCP/IP

---

## client\_userid - Client user ID monitor element

The client user ID generated by a transaction manager and provided to the server, if the sqleseti API is used. The current value of the CLIENT\_USERID special register for this connection, unit of work, or activity.

**Note:** This element is reported for the coordinating member only. On remote members, the value reported is a string with a length of 0.

This monitor element is synonymous to the **tpmon\_client\_userid** monitor element. The **client\_userid** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_userid** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

*Table 362. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 363. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Locking        | -                     | -                                |
| Unit of work   | -                     | -                                |
| Change history | changesummary         | Always collected                 |

### Usage

Use this element in application server or Transaction Processing monitor environments to identify the end-user for whom the transaction is being executed.

---

## client\_wrkstname - Client workstation name monitor element

Identifies the client's system or workstation (for example CICS EITERMID), if the sqleseti API was issued in this connection. The current value of the CLIENT\_WRKSTNNAME special register for this connection, unit of work, or activity.

**Note:** This element is reported for the coordinating member only. On remote members, the value reported is a string with a length of 0.

This monitor element is synonymous to the **tpmon\_client\_wkstn** monitor element. The **client\_wrkstnname** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_wkstn** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

*Table 364. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                              | ACTIVITY METRICS BASE            |

*Table 365. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Locking        | -                     | -                                |
| Unit of work   | -                     | -                                |
| Change history | changesummary         | Always collected                 |

## Usage

Use this element to identify the user's machine by node ID, terminal ID, or similar identifiers.

---

## codepage\_id - ID of Code Page Used by Application

The code page identifier.

*Table 366. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 367. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |
| Connections      | event_connheader      | Always collected |

**Usage** For snapshot monitor data, this is the code page at the partition where the monitored application started. This identifier may be used for problem determination for remote applications. You may use this information to ensure that data conversion is supported between the application code page and the database code page (or for DRDA host databases, the host CCSID). For information about supported code pages, see the *Administration Guide* .

For event monitor data, this is the code page of the database for which event data is collected. You can use this element to determine whether your event monitor application is running under a different code page from that used by the database. Data written by the event monitor uses the database code page. If your event monitor application uses a different code page, you may need to perform some character conversion to make the data readable.

---

## comm\_exit\_wait\_time - Communication buffer exit wait time monitor element

Time spent waiting for the return from a communication buffer exit library API function. The value is given in milliseconds.

Table 368. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics                              | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics             | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                          | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics                                  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                 | REQUEST METRICS BASE             |

Table 369. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Unit of work | Reported in the system_metrics document.              | Always collected     |
| Statistics   | Event_scstats (reported in the details_xml document). | REQUEST METRICS BASE |
| Statistics   | Event_wlstats (reported in the details_xml document). | REQUEST METRICS BASE |

## comm\_exit\_waits - Communication buffer exit number of waits monitor element

The number of times a communication buffer exit library API function is called.

Table 370. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics                              | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics             | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                          | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics                                  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                 | REQUEST METRICS BASE             |

Table 371. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Unit of work | Reported in the system_metrics document.              | Always collected     |
| Statistics   | Event_scstats (reported in the details_xml document). | REQUEST METRICS BASE |
| Statistics   | Event_wlstats (reported in the details_xml document). | REQUEST METRICS BASE |

---

## comm\_private\_mem - Committed Private Memory

The amount of private memory that the instance of the database manager has currently committed at the time of the snapshot. The comm\_private\_mem value returned is only relevant on Windows operating systems.

Table 372. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

---

## commit\_sql\_stmts - Commit Statements Attempted

The total number of SQL COMMIT statements that have been attempted.

Table 373. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Database        | dbase                 | Basic          |
| Database        | dbase_remote          | Basic          |
| Application     | appl                  | Basic          |
| Application     | appl_remote           | Basic          |
| DCS Database    | dcs_dbase             | Basic          |
| DCS Application | dcs_appl              | Basic          |

For snapshot monitoring, this counter can be reset.

Table 374. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** A small rate of change in this counter during the monitor period may indicate that applications are not doing frequent commits, which may lead to problems with logging and data concurrency.

You can also use this element to calculate the total number of units of work using the following expression:

```
commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

**Note:** The units of work calculated will only include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at a database or application level.

---

## comp\_env\_desc - Compilation environment monitor element

This element stores information about the compilation environment used when compiling the SQL statement.

*Table 375. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries  | ACTIVITY METRICS BASE            |

*Table 376. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch    |
|--|-----------------------|-------------------|
| Deadlocks with Details<br>History Values | event_stmt_history    | Always collected  |
| Deadlocks with Details<br>History        | event_stmt_history    | Always collected  |
| Activities                               | event_activitystmt    | Always collected  |
| Package cache                            | -                     | COLLECT BASE DATA |

*Table 377. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

### Usage

This monitor element stores the compilation environment description in a binary large object. To view this information in readable form, use the COMPILATION\_ENV table function.

You can provide this element as input to the COMPILATION\_ENV table function, or to the SET COMPILATION ENVIRONMENT SQL statement.

---

## completion\_status - Completion status monitor element

The status of the unit of work.

*Table 378. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |

### Usage

Use this element to determine if the unit of work ended due to a deadlock or abnormal termination. The possible values are listed in the sqllib/misc/DB2EvmonUOW.xsd file:

- UNKNOWN

- COMMIT
- ROLLBACK
- GLOBAL\_COMMIT
- GLOBAL\_ROLLBACK
- XA\_END
- XA\_PREPARE

---

## configured\_cf\_gbp\_size - Configured cluster caching facility group buffer pool size monitor element

The allocated and reserved group buffer pool memory specified using the **cf\_gbp\_sz** configuration parameter, in pages with a page size of 4 KB.

*Table 379. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## configured\_cf\_lock\_size - Configured cluster caching facility lock size monitor element

Global lock memory configured, in pages with a page size of 4 KB. This value is specified using the **cf\_lock\_sz** configuration parameter.

*Table 380. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## configured\_cf\_sca\_size - Configured cluster caching facility shared communications area size monitor element

Shared communications area memory currently allocated and reserved, in pages with a page size of 4 KB. This value is specified using the **cf\_sca\_sz** configuration parameter.

*Table 381. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## configured\_cf\_mem\_size - Configured cluster caching facility memory size monitor element

Configured total memory size for the cluster caching facility, in pages with a page size of 4 KB. This value is specified using the **cf\_mem\_sz** configuration parameter.

*Table 382. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## con\_elapsed\_time - Most Recent Connection Elapsed Time

The elapsed time that the DCS application that most recently disconnected from this host database was connected.

### Element identifier

con\_elapsed\_time

### Element type

time

*Table 383. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Timestamp      |

## Usage

Use this element as an indicator of the length of time that applications are maintaining connections to a host database.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## con\_local\_databases - Local Databases with Current Connects

The number of local databases that have applications connected.

*Table 384. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

**Usage** This value gives an indication of how many database information records you can expect when gathering data at the database level.

The applications can be running locally or remotely, and may or may not be executing a unit of work within the database manager

---

## con\_response\_time - Most Recent Response Time for Connect

The elapsed time between the start of connection processing and actual establishment of a connection, for the most recent DCS application that connected to this database.

### Element identifier

con\_response\_time

### Element type

time

Table 385. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Timestamp      |

## Usage

Use this element as an indicator of the time it currently takes applications to connect to a particular host database.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## concurrent\_act\_top - Concurrent activity top monitor element

The high watermark for the concurrent activities (at any nesting level) in a service subclass since the last reset.

**Note:** This element monitors the highest concurrent execution of all activities, including those activities that do not participate in the CONCURRENTDBCOORDACTIVITIES threshold. For example, although CALL statements do not count toward the concurrency that is enforced by the CONCURRENTDBCOORDACTIVITIES threshold, they are included in the concurrent activity high watermark measurement.

Table 386. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | ACTIVITY METRICS BASE            |

Table 387. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

## Usage

Use this element to know the highest concurrency of activities (including nested activities) reached on a member for a service subclass in the time interval collected.

---

## concurrent\_connection\_top - Concurrent connection top monitor element

High watermark for concurrent coordinator connections in this service class since the last reset. This field has the same value in every subclass of the same superclass.

*Table 388. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUPERCLASS_STATS<br>table function - Return statistics of service superclasses | ACTIVITY METRICS BASE            |

*Table 389. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

## Usage

This element may be useful in determining where to place thresholds on connection concurrency by showing where the current high watermark is. It is also useful for verifying that such a threshold is configured correctly and doing its job.

---

## concurrent\_wlo\_act\_top - Concurrent WLO activity top monitor element

High watermark for concurrent activities (at any nesting level) of any occurrence of this workload since the last reset.

*Table 390. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics | ACTIVITY METRICS BASE            |

*Table 391. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |

## Usage

Use this element to know the highest number of concurrent activities reached on a member for any occurrence of this workload in the time interval collected.

---

## concurrent\_wlo\_top - Concurrent workload occurrences top monitor element

The high watermark for the concurrent occurrences of a workload since the last reset.

*Table 392. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | ACTIVITY METRICS BASE            |

*Table 393. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |
| Statistics | event_scstats         | -              |

### Usage

Use this element to know the highest concurrency of workload occurrences reached on a member for a workload in the time interval collected.

---

## concurrentdbcoordactivities\_db\_threshold\_id - Concurrent database coordinator activities database threshold ID monitor element

The ID of the CONCURRENTDBCOORDACTIVITIES database threshold that was applied to the activity.

*Table 394. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand which CONCURRENTDBCOORDACTIVITIES database threshold, if any, was applied to the activity.

---

## concurrentdbcoordactivities\_db\_threshold\_queued - Concurrent database coordinator activities database threshold queued monitor element

This monitor element returns 'Yes' to indicate that the activity was queued by the CONCURRENTDBCOORDACTIVITIES database threshold. 'No' indicates that the activity was not queued.

*Table 395. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand if the activity was queued by the CONCURRENTDBCOORDACTIVITIES database threshold applied to the activity.

---

## concurrentdbcoordactivities\_db\_threshold\_value - Concurrent database coordinator activities database threshold value monitor element

This monitor element returns the upper bound of the CONCURRENTDBCOORDACTIVITIES database threshold that was applied to the activity.

*Table 396. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand the value of the CONCURRENTDBCOORDACTIVITIES database threshold applied to the activity, if any.

---

## **concurrentdbcoordactivities\_db\_threshold\_violated - Concurrent database coordinator activities database threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CONCURRENTDBCOORDACTIVITIES database threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 397. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                   |

### **Usage**

Use this element to determine if the activity violated the CONCURRENTDBCOORDACTIVITIES database threshold that was applied to the activity.

---

## **concurrentdbcoordactivities\_subclass\_threshold\_id - Concurrent database coordinator activities service subclass threshold ID monitor element**

This monitor element returns the ID of the CONCURRENTDBCOORDACTIVITIES service subclass threshold that was applied to the activity.

*Table 398. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                   |

### **Usage**

Use this element to understand which CONCURRENTDBCOORDACTIVITIES service subclass threshold, if any, was applied to the activity.

---

## **concurrentdbcoordactivities\_subclass\_threshold\_queued - Concurrent database coordinator activities service subclass threshold queued monitor element**

This monitor element returns 'Yes' to indicate that the activity was queued by the CONCURRENTDBCOORDACTIVITIES service subclass threshold. 'No' indicates that the activity was not queued.

*Table 399. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                   |

## Usage

Use this element to understand if the activity was queued by the CONCURRENTDBCOORDACTIVITIES service subclass threshold applied to the activity.

---

### **concurrentdbcoordactivities\_subclass\_threshold\_value - Concurrent database coordinator activities service subclass threshold value monitor element**

This monitor element returns the upper bound of the CONCURRENTDBCOORDACTIVITIES service subclass threshold that was applied to the activity.

*Table 400. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_ACTIVITY_DETAILS table function - Return detailed information about a specific activity            | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the CONCURRENTDBCOORDACTIVITIES service subclass threshold applied to the activity, if any.

---

### **concurrentdbcoordactivities\_subclass\_threshold\_violated - Concurrent database coordinator activities service subclass threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CONCURRENTDBCOORDACTIVITIES service subclass threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 401. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the CONCURRENTDBCOORDACTIVITIES service subclass threshold that was applied to the activity.

---

## concurrentdbcoordactivities\_superclass\_threshold\_id - Concurrent database coordinator activities service superclass threshold ID monitor element

The ID of the CONCURRENTDBCOORDACTIVITIES\_SUPERCLASS threshold that was applied to the activity.

*Table 402. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand which CONCURRENTDBCOORDACTIVITIES service superclass threshold, if any, was applied to the activity.

---

## concurrentdbcoordactivities\_superclass\_threshold\_queued - Concurrent database coordinator activities service superclass threshold queued monitor element

This monitor element returns 'Yes' to indicate that the activity was queued by the CONCURRENTDBCOORDACTIVITIES service superclass threshold. 'No' indicates that the activity was not queued.

*Table 403. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand if the activity was queued by the CONCURRENTDBCOORDACTIVITIES service superclass threshold applied to the activity.

---

## concurrentdbcoordactivities\_superclass\_threshold\_value - Concurrent database coordinator activities service superclass threshold value monitor element

The upper bound of the CONCURRENTDBCOORDACTIVITIES service superclass threshold that was applied to the activity.

*Table 404. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the CONCURRENTDBCOORDACTIVITIES service superclass threshold applied to the activity, if any.

---

### **concurrentdbcoordactivities\_superclass\_threshold\_violated - Concurrent database coordinator activities service superclass threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CONCURRENTDBCOORDACTIVITIES service superclass threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 405. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the CONCURRENTDBCOORDACTIVITIES service superclass threshold that was applied to the activity.

---

### **concurrentdbcoordactivities\_wl\_was\_threshold\_id - Concurrent database coordinator activities workload work action set threshold ID monitor element**

The identifier of the CONCURRENTDBCOORDACTIVITIES workload work action set threshold that was applied to the activity.

*Table 406. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which CONCURRENTDBCOORDACTIVITIES workload work action set threshold, if any, was applied to the activity.

---

## concurrentdbcoordactivities\_wl\_was\_threshold\_queued - Concurrent database coordinator activities workload work action set threshold queued monitor element

This monitor element returns 'Yes' to indicate that the activity was queued by the CONCURRENTDBCOORDACTIVITIES workload work action set threshold. 'No' indicates that the activity was not queued.

*Table 407. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand if the activity was queued by the CONCURRENTDBCOORDACTIVITIES workload work action set threshold applied to the activity.

---

## concurrentdbcoordactivities\_wl\_was\_threshold\_value - Concurrent database coordinator activities workload work action set threshold value monitor element

The upper bound of the CONCURRENTDBCOORDACTIVITIES workload work action set threshold that was applied to the activity.

*Table 408. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand the value of the CONCURRENTDBCOORDACTIVITIES workload work action set threshold applied to the activity.

---

## concurrentdbcoordactivities\_wl\_was\_threshold\_violated - Concurrent database coordinator activities workload work action set threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the CONCURRENTDBCOORDACTIVITIES workload work action set threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 409. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to determine if the activity violated the CONCURRENTDBCOORDACTIVITIES workload work action set threshold that was applied to the activity.

---

## concurrentdbcoordactivities\_work\_action\_set\_threshold\_id - Concurrent database coordinator activities work action set threshold ID monitor element

The ID of the CONCURRENTDBCOORDACTIVITIES work action set threshold that was applied to the activity.

*Table 410. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand which CONCURRENTDBCOORDACTIVITIES work action set threshold, if any, was applied to the activity.

---

## concurrentdbcoordactivities\_work\_action\_set\_threshold\_queued - Concurrent database coordinator activities work action set threshold queued monitor element

This monitor element returns 'Yes' to indicate that the activity was queued by the CONCURRENTDBCOORDACTIVITIES\_WORK\_ACTION\_SET threshold. 'No' indicates that the activity was not queued.

*Table 411. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand if the activity was queued by the CONCURRENTDBCOORDACTIVITIES\_WORK\_ACTION\_SET threshold applied to the activity.

---

### **concurrentdbcoordactivities\_work\_action\_set\_threshold\_value - Concurrent database coordinator activities work action set threshold value monitor element**

The upper bound of the CONCURRENTDBCOORDACTIVITIES\_WORK\_ACTION\_SET threshold that was applied to the activity.

*Table 412. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the CONCURRENTDBCOORDACTIVITIES\_WORK threshold applied to the activity, if any.

---

### **concurrentdbcoordactivities\_work\_action\_set\_threshold\_violated - Concurrent database coordinator activities work action set threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CONCURRENTDBCOORDACTIVITIES\_WORK\_ACTION\_SET threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 413. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the CONCURRENTDBCOORDACTIVITIES\_WORK\_ACTION\_SET threshold that was applied to the activity.

---

### **conn\_complete\_time - Connection Request Completion Timestamp**

The date and time that a connection request was granted.

**Element identifier**

conn\_complete\_time

**Element type**  
timestamp

*Table 414. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Timestamp      |

**Usage** Use this element to determine when a connection request to the database was granted.

---

## conn\_time - Time of database connection monitor element

The date and time of the connection to the database (at the database level, this is the first connection to the database), or when the activate database was issued.

*Table 415. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |
| Database     | event_dbheader        | Always collected |
| Connections  | event_connheader      | Always collected |

### Usage

Use this element with the **disconn\_time** monitor element to calculate the elapsed time since:

- The database was active (for information at the database level).
- The connection was active (for information at the connection level).

---

## connection\_start\_time - Connection start time monitor element

The time at which the connection was established with the database server. The connection\_time monitor element is an alias of the connection\_start\_time monitor element.

*Table 416. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

---

## connection\_status - Connection Status

For snapshot monitor, this monitor element reports the status of the communication connection between the node issuing the GET SNAPSHOT command and other nodes listed in the db2nodes.cfg file. For table function monitor, this monitor element reports the text identifier indicating the FCM connection status.

Table 417. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

Table 418. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

## Usage

For the snapshot monitor, the connection values are :

**SQLM\_FCM\_CONNECT\_INACTIVE**

No current connection

**SQLM\_FCM\_CONNECT\_ACTIVE**

Connection is active

For table function monitoring, the available values are:

**Active** No current connection

**Inactive**

Connection is active

Two members can be active, but the communication connection between them remains inactive until there is some communication between the members.

---

## connections\_top - Maximum Number of Concurrent Connections

The highest number of simultaneous connections to the database since the database was activated.

**Element identifier**

connections\_top

**Element type**

watermark

Table 419. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 420. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You may use this element to evaluate the setting of the *maxappls* configuration parameter.

If the value of this element is the same as the *maxappls* parameter, it is likely that some database connection requests were rejected, since *maxappls* limits the number of database connections allowed.

The current number of connections at the time the snapshot was taken can be calculated using the following formula:

$$\text{rem\_cons\_in} + \text{local\_cons}$$


---

## consistency\_token - Package consistency token monitor element

For a given package name and creator, there can exist (starting in DB2 Version 8) multiple versions. The package consistency token helps to identify the version of the package that contains the SQL currently executing.

*Table 421. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

*Table 422. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Locking    | -                     | Always collected |
| Statements | event_stmt            | Always collected |

### Usage

You can use this element to help identify the package and the SQL statement that is executing.

---

## container\_accessible - Accessibility of container monitor element

This element indicates whether a container is accessible. A value of 1 means "Yes"; a value of 0 means "No".

*Table 423. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONTAINER table function -<br>Get table space container metrics | ACTIVITY METRICS BASE            |

*Table 424. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_container  | Basic          |

**Usage** This element can be used in conjunction with the elements **container\_id**, **container\_name**, **container\_type**, **container\_total\_pages**, **container\_usable\_pages**, and **container\_stripe\_set** to describe the container.

---

## container\_id - Container identification monitor element

An integer that uniquely defines a container within a table space.

Table 425. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get container metrics                               | ACTIVITY METRICS BASE            |

Table 426. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_container  | Basic          |

**Usage** This element can be used in conjunction with the elements container\_name, container\_type, container\_total\_pages, container\_usable\_pages, container\_stripe\_set, and container\_accessible to describe the container.

---

## container\_name - Container name monitor element

The name of a container.

Table 427. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONTAINER table function - Get container metrics | ACTIVITY METRICS BASE            |

Table 428. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_container  | Basic          |

**Usage** This element can be used in conjunction with the elements container\_id, container\_type, container\_total\_pages, container\_usable\_pages, container\_stripe\_set, and container\_accessible to describe the container.

---

## container\_stripe\_set - Container stripe set monitor element

The stripe set that a container belongs to.

Table 429. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_CONTAINER table function - Get table space container metrics | ACTIVITY METRICS BASE                        |

Table 430. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_container  | Basic          |

## Usage

Use this monitor element in conjunction with the elements **container\_id**, **container\_name**, **container\_type**, **container\_total\_pages**, **container\_usable\_pages**, and **container\_accessible** to describe the container. This is only applicable to a DMS table space.

---

## container\_total\_pages - Total pages in container monitor element

The total number of pages occupied by the container.

*Table 431. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_CONTAINER table function - Get table space container metrics | ACTIVITY METRICS BASE                        |

*Table 432. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|--|
| Table Space    | tablespace_container  | Basic (DMS table spaces)<br>Buffer Pool (SMS table spaces) |

**Usage** This element can be used in conjunction with the elements **container\_id**, **container\_name**, **container\_type**, **container\_usable\_pages**, **container\_stripe\_set**, and **container\_accessible** to describe the container.

---

## container\_type - Container type monitor element

The type of the container.

*Table 433. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONTAINER table function - Get container metrics | ACTIVITY METRICS BASE            |

*Table 434. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_container  | Basic          |

## Usage

This element returns the type of the container, which can be a directory path (for SMS only), file (for DMS) or a raw device (for DMS). This element can be used in conjunction with the elements **container\_id**, **container\_name**, **container\_total\_pages**, **container\_usable\_pages**, **container\_stripe\_set**, and **container\_accessible** to describe the container.

The valid values for this monitor element are defined in the `sqlutil.h` file.

---

## container\_usable\_pages - Usable pages in container monitor element

The total number of usable pages in a container.

Table 435. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_CONTAINER table function - Get table space container metrics | ACTIVITY METRICS BASE                        |

Table 436. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|--|
| Table Space    | tablespace_container  | Basic (DMS table spaces)<br>Buffer Pool (SMS table spaces) |

**Usage** This element can be used in conjunction with the elements container\_id, container\_name, container\_type, container\_total\_pages, container\_stripe\_set, and container\_accessible to describe the container. For SMS table spaces, this value is the same as container\_total\_pages.

---

## coord\_act\_aborted\_total - Coordinator activities aborted total monitor element

The total number of coordinator activities at any nesting level that completed with errors since the last reset. For service classes, the value is updated when the activity completes. For workloads, the value is updated by each workload occurrence at the end of its unit of work.

For service classes, if you remap an activity to a different subclass with a REMAP ACTIVITY action before it aborts, then this activity counts only toward the total of the subclass it aborts in.

Table 437. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics                      | ACTIVITY METRICS BASE            |

Table 438. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wlstats         | -              |

## Usage

Use this element to understand if activities on the system are completing successfully. Activities may be aborted due to cancellation, errors or reactive thresholds.

---

### **coord\_act\_completed\_total - Coordinator activities completed total monitor element**

The total number of coordinator activities at any nesting level that completed successfully since the last reset. For service classes, the value is updated when the activity completes. For workloads, the value is updated by each workload occurrence at the end of its unit of work.

For service classes, if you remap an activity to a different subclass with a REMAP ACTIVITY action before it completes, then this activity counts only toward the total of the subclass it completes in.

*Table 439. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics                      | ACTIVITY METRICS BASE            |

*Table 440. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |
| Statistics | event_scstats         | -              |

## Usage

This element can be used to determine the throughput of activities in the system or to aid in calculating average activity lifetime across multiple members.

---

### **coord\_act\_est\_cost\_avg - Coordinator activity estimated cost average monitor element**

Arithmetic mean of the estimated costs for coordinator DML activities at nesting level 0 associated with this service subclass or work class since the last reset. If the internally tracked average has overflowed, the value -2 is returned. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE or BASE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA EXTENDED work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE or BASE. Units are timerons.

For service classes, the estimated cost of an activity is counted only toward the service subclass in which the activity enters the system. When you remap activities between service subclasses with a REMAP ACTIVITY action, the coord\_act\_est\_cost\_avg mean of the service subclass you remap an activity to is unaffected.

Table 441. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_scstats         | Always collected |
| Statistics | event_wcstats         | Always collected |
| Statistics | event_wlstats         | Always collected |

## Usage

Use this statistic to determine the arithmetic mean of the estimated costs of coordinator DML activities at nesting level 0 that are associated this service subclass, workload, or work class that completed or aborted since the last statistics reset.

This average can also be used to determine whether or not the histogram template used for the activity estimated cost histogram is appropriate. Compute the average activity estimated cost from the activity estimated cost histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template for the activity estimated cost histogram, using a set of bin values that are more appropriate for your data.

---

## coord\_act\_exec\_time\_avg - Coordinator activities execution time average monitor element

Arithmetic mean of execution times for coordinator activities at nesting level 0 associated with this service subclass or work class since the last reset. If the internally tracked average has overflowed, the value -2 is returned. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE. Units are milliseconds.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action, the coord\_act\_exec\_time\_avg mean of service subclasses an activity is mapped to but does not complete in is unaffected.

Table 442. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | COLLECT AGGREGATE ACTIVITY DATA  |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | COLLECT AGGREGATE ACTIVITY DATA  |

Table 443. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_wlstats         | -              |

## Usage

Use this statistic to determine the arithmetic mean of execution time for coordinator activities associated with a service subclass, workload, or work class that completed or aborted.

This average can also be used to determine whether or not the histogram template used for the activity execution time histogram is appropriate. Compute the average activity execution time from the activity execution time histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template for the activity execution time histogram, using a set of bin values that are more appropriate for your data.

---

## coord\_act\_interarrival\_time\_avg - Coordinator activity arrival time average monitor element

Arithmetic mean of the time between arrivals of coordinator activities at nesting level 0 associated with this service subclass or work class since the last reset. If the internally tracked average has overflowed, the value -2 is returned. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE or BASE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA EXTENDED work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE or BASE. Units are milliseconds.

For service classes, the inter-arrival time mean is calculated for service subclasses through which activities enter the system. When you remap activities between service subclasses with a REMAP ACTIVITY action, the coord\_act\_interarrival\_time\_avg of the service subclass you remap an activity to is unaffected.

Table 444. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_wlstats         | -              |

## Usage

Use this statistic to determine the arithmetic mean between arrivals of coordinator activities at nesting level 0 associated with this service subclass, workload, or work class.

The inter-arrival time can be used to determine arrival rate, which is the inverse of inter-arrival time. This average can also be used to determine whether or not the histogram template used for the activity inter-arrival time histogram is appropriate. Compute the average activity inter-arrival time from the activity inter-arrival time histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template for the activity inter-arrival time histogram, using a set of bin values that are more appropriate for your data.

---

## **coord\_act\_lifetime\_avg - Coordinator activity lifetime average monitor element**

Arithmetic mean of lifetime for coordinator activities at nesting level 0 associated with this service subclass, workload, or work class since the last reset. If the internally tracked average has overflowed, the value -2 is returned. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE. Units are milliseconds.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action, only the the coord\_act\_lifetime\_avg mean of the final service class where the activity completes is affected.

*Table 445. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | COLLECT AGGREGATE ACTIVITY DATA         |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | COLLECT AGGREGATE ACTIVITY DATA         |

*Table 446. Event Monitoring Information*

| <b>Event Type</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-------------------|------------------------------|-----------------------|
| Statistics        | event_scstats                | -                     |
| Statistics        | event_wcstats                | -                     |
| Statistics        | event_wlstats                | -                     |

### **Usage**

Use this statistic to determine the arithmetic mean of the lifetime for coordinator activities associated with a service subclass, workload, or work class that completed or aborted.

This statistic can also be used to determine whether or not the histogram template used for the activity lifetime histogram is appropriate. Compute the average activity lifetime from the activity lifetime histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template

for the activity lifetime histogram, using a set of bin values that are more appropriate for your data.

---

## coord\_act\_lifetime\_top - Coordinator activity lifetime top monitor element

High watermark for coordinator activity lifetime, counted at all nesting levels. Units are milliseconds. For service classes, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE. For work classes, this monitor element returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE.

To effectively use this statistic with service classes when you also remap activities between service subclasses with a REMAP ACTIVITY action, you must aggregate the coord\_act\_lifetime\_top high watermark of any given service subclass with that of other subclasses affected by the same remapping threshold or thresholds. This is because an activity will complete after it has been remapped to a different service subclass by a remapping threshold, and the time the activity spends in other service subclasses before being remapped is counted only toward the service class in which it completes.

*Table 447. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | COLLECT AGGREGATE ACTIVITY DATA  |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | COLLECT AGGREGATE ACTIVITY DATA  |

*Table 448. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wcstats         | -              |
| Statistics | event_scstats         | -              |
| Statistics | event_wlstats         | -              |

### Usage

This element can be used to help determine whether or not thresholds on activity lifetime are being effective and can also help to determine how to configure such thresholds.

---

## coord\_agent\_tid - Coordinator agent engine dispatchable unit ID monitor element

The engine dispatchable unit (EDU) identifier of the coordinator agent for the application.

Table 449. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## coord\_act\_queue\_time\_avg - Coordinator activity queue time average monitor element

Arithmetic mean of queue time for coordinator activities at nesting level 0 associated with this service subclass or work class since the last reset. If the internally tracked average has overflowed, the value -2 is returned. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE. For work classes, this monitor element returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE. Units are milliseconds.

For service classes, the queue time counts only toward the service subclass in which the activity completes or is aborted. When you remap activities between service subclasses with a REMAP ACTIVITY action, the coord\_act\_queue\_time\_avg mean of service subclasses an activity is mapped to but does not complete in is unaffected.

Table 450. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | COLLECT AGGREGATE ACTIVITY DATA  |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics                      | COLLECT AGGREGATE ACTIVITY DATA  |

Table 451. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_scstats         | Always collected |
| Statistics | event_wcstats         | Always collected |
| Statistics | event_wlstats         | Always collected |

### Usage

Use this statistic to determine the arithmetic mean of the queue time for coordinator activities associated with a service subclass, workload, or work class that completed or aborted.

This statistic can also be used to determine whether or not the histogram template used for the activity queue time histogram is appropriate. Compute the average

activity queue time from the activity queue time histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template for the activity queue time histogram, using a set of bin values that are more appropriate for your data.

---

## coord\_act\_rejected\_total - Coordinator activities rejected total monitor element

The total number of coordinator activities at any nesting level that were rejected instead of being allowed to execute since the last reset. This counter is updated when an activity is prevented from executing by either a predictive threshold or a prevent execution work action. For service classes, the value is updated when the activity completes. For workloads, the value is updated by each workload occurrence at the end of its unit of work.

*Table 452. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | COLLECT AGGREGATE ACTIVITY DATA  |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics                      | ACTIVITY METRICS BASE            |

*Table 453. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wlstats         | -              |

### Usage

This element can be used to help determine whether or not predictive thresholds and work actions that prevent execution are being effective and whether or not they are too restrictive.

---

## coord\_agent\_pid - Coordinator agent identifier monitor element

The engine dispatchable unit (EDU) identifier of the coordinator agent for the application. Except on the Linux operating system, the EDU ID is mapped to the thread ID. On the Linux operating system, the EDU ID is a DB2 generated unique identifier.

*Table 454. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |

Table 455. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | -                     | -              |

## Usage

You can use this element to link database system monitor information to other sources of diagnostic information, such as system traces.

---

## coord\_agents\_top - Maximum Number of Coordinating Agents

The maximum number of coordinating agents working at one time.

Table 456. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| Database         | dbase                 | Basic          |

## Usage

If the peak number of coordinating agents represents too high a workload for this node, you can reduce this upper boundary by changing the **max\_coordagents** configuration parameter.

---

## coord\_member - Coordinator member monitor element

Coordinating member for an application.

Table 457. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                                | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                          | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                            | ACTIVITY METRICS BASE            |

Table 458. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Unit of work   | -                     | Always collected |
| Change history | changesummary         | Always collected |

## coord\_node - Coordinating Node

In a multi-node system, the node number of the node where the application connected or attached to the instance.

Table 459. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 460. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** Each connected application is served by one coordinator node.

## coord\_partition\_num - Coordinator partition number monitor element

The coordinator partition of the unit of work or activity. In a multi-partition system, the coordinator partition is the partition where the application connected to the database.

Table 461. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities   | ACTIVITY METRICS BASE            |

Table 462. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch   |
|----------------------|---------------------------|------------------|
| Unit of work         | -                         | Always collected |
| Activities           | event_activity            | Always collected |
| Threshold violations | event_thresholdviolations | Always collected |

### Usage

This element allows the coordinator partition to be identified for activities or units of work that have records on partitions other than the coordinator.

---

## coord\_stmt\_exec\_time - Execution time for statement by coordinator agent monitor element

The total time spent executing this statement by coordinator agents on this member. The value is given in milliseconds.

Table 463. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE  |

Table 464. Event Monitoring Information

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activitymetrics                      | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

---

## corr\_token - DRDA Correlation Token

The DRDA AS correlation token.

Table 465. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Application    | appl                  | Basic          |

Table 466. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Connections | event_connheader      | Always collected |

**Usage** The DRDA correlation token is used for correlating the processing between the application server and the application requester. It is an identifier dumped into logs when errors arise, that you can use to identify the conversation that is in error. In some cases, it will be the LUWID of the conversation.

If communications are not using DRDA, this element returns the *appl\_id* (see *appl\_id*).

If you are using the database system monitor APIs, note that the API constant `SQLM_APPLID_SZ` is used to define the length of this element.

---

## cost\_estimate\_top - Cost estimate top monitor element

The high watermark for the estimated cost of DML activities at all nesting levels in a service subclass or work class. For service subclasses, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service subclass is set to NONE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class.

For service classes, the estimated cost of DML activities is counted only toward the service subclass in which the activity enters the system. When you remap activities between service subclasses with a REMAP ACTIVITY action, the cost\_estimate\_top of the service subclass you remap an activity to is unaffected.

Table 467. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_wlstats         | -              |

### Usage

Use this element to determine the highest DML activity estimated cost reached on a member for a service class, workload, or work class in the time interval collected.

---

## count - Number of Event Monitor Overflows

The number of consecutive overflows that have occurred.

### Element identifier

count

### Element type

counter

Table 468. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based<br>component times                   | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW -<br>Get formatted row-based combined hierarchy<br>wait and processing times | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY<br>_ROW - Get formatted row-based output for<br>wait times                  | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |

Table 469. Event Monitoring Information

| Event Type      | Logical Data Grouping | Monitor Switch   |
|-----------------|-----------------------|------------------|
| Overflow Record | event_overflow        | Always collected |

**Usage** You may use this element to get an indication of how much monitor data has been lost.

The event monitor sends one overflow record for a set of consecutive overflows.

---

## cpu\_configured - Number of configured CPUs monitor element

The number of processors on this host that the operating system is aware of.

*Table 470. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_cores\_per\_socket - Number of CPU cores per socket monitor element

The number of processors on this host. On single core systems, this value is 1.

*Table 471. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_hmt\_degree - Number of logical CPUs monitor element

On systems that support hardware multithreading, the number of logical processors that appear to be present as a result of multithreading. On systems that do not support multithreading, this value is 1.

*Table 472. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_idle - Processor idle time monitor element

Processor idle time, expressed in processor ticks. Reported for Windows, AIX and Linux systems only. This measurement represents the aggregate for all processors on the system.

*Table 473. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

### Usage

- This measurement represents the aggregate for all processors on the system.
- On AIX, this metric is reported for the workload partition (WPAR) and the logical partition (LPAR) on which the DB2 server is running.

- You can use this monitor element along with the related processor timer elements to calculate processor utilization for a specific time interval on the host system. To calculate processor utilization as a percentage, perform the following steps:
  1. Use the ENV\_GET\_SYSTEM\_RESOURCES function at the beginning to the time interval to retrieve the values for the following metrics:
    - `cpu_usert1` = **cpu\_user**
    - `cpu_systemt1` = **cpu\_system**
    - `cpu_idlet1` = **cpu\_idle**
    - `cpu_waitt1` = **cpu\_wait**
  2. Repeat the preceding step to determine the timestamps for the same metrics at the end of the time interval for which you want to calculate processor utilization:
    - `cpu_usert2` = **cpu\_user**
    - `cpu_systemt2` = **cpu\_system**
    - `cpu_idlet2` = **cpu\_idle**
    - `cpu_iowaitt2` = **cpu\_iowait**
  3. Calculate processor utilization using the following formula:
 
$$100 \times \frac{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1})}{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1}) + (\text{cpu\_idle}_{t2} - \text{cpu\_idle}_{t1}) + (\text{cpu\_iowait}_{t2} - \text{cpu\_iowait}_{t1})}$$

---

## cpu\_iowait - IO Wait time monitor element

Time spent waiting for IO (Linux, UNIX); time spent receiving and servicing hardware interrupts (Windows), expressed in processor ticks. Reported for Windows, AIX and Linux systems only. This measurement represents the aggregate for all processors on the system.

*Table 474. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table function - Return system information | ACTIVITY METRICS BASE            |

### Usage

- This measurement represents the aggregate for all processors on the system.
- On AIX, this metric is reported for the workload partition (WPAR) and the logical partition (LPAR) on which the DB2 server is running.
- You can use this monitor element along with the related processor timer elements to calculate processor utilization for a specific time interval on the host system. To calculate processor utilization as a percentage, perform the following steps:
  1. Use the ENV\_GET\_SYSTEM\_RESOURCES function at the beginning to the time interval to retrieve the values for the following metrics:
    - `cpu_usert1` = **cpu\_user**
    - `cpu_systemt1` = **cpu\_system**
    - `cpu_idlet1` = **cpu\_idle**
    - `cpu_waitt1` = **cpu\_wait**

2. Repeat the preceding step to determine the timestamps for the same metrics at the end of the time interval for which you want to calculate processor utilization:
  - $\text{cpu\_user}_{t_2} = \text{cpu\_user}$
  - $\text{cpu\_system}_{t_2} = \text{cpu\_system}$
  - $\text{cpu\_idle}_{t_2} = \text{cpu\_idle}$
  - $\text{cpu\_iowait}_{t_2} = \text{cpu\_iowait}$
3. Calculate processor utilization using the following formula:
 
$$100 \times \frac{(\text{cpu\_system}_{t_2} - \text{cpu\_system}_{t_1}) + (\text{cpu\_user}_{t_2} - \text{cpu\_user}_{t_1})}{(\text{cpu\_system}_{t_2} - \text{cpu\_system}_{t_1}) + (\text{cpu\_user}_{t_2} - \text{cpu\_user}_{t_1}) + (\text{cpu\_idle}_{t_2} - \text{cpu\_idle}_{t_1}) + (\text{cpu\_iowait}_{t_2} - \text{cpu\_iowait}_{t_1})}$$

## cpu\_limit - WLM dispatcher CPU limit monitor element

The WLM dispatcher CPU limit configured for the service class.

Table 475. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics | ACTIVITY METRICS BASE            |

## cpu\_load\_long - Processor load (long timeframe) monitor element

Processor load over the longer term, as defined by the system. For example, the average processor load over the past 10 or 15 minutes. Reported for all platforms except Windows.

Table 476. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

## cpu\_load\_medium - Processor load (medium timeframe) monitor element

Processor load over the medium term, as defined by the system. For example, the average processor load over the past 5 or 10 minutes. Reported for all platforms except Windows.

Table 477. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_load\_short - Processor load (short timeframe) monitor element

Processor load over the short term, as defined by the system. For example, the average processor load over the past 1 or 5 minutes. Reported for all platforms except Windows.

*Table 478. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_online - Number of CPUs online monitor element

The number of processors on this host that are currently online.

*Table 479. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_share\_type - WLM dispatcher CPU share type monitor element

The type of WLM dispatcher CPU shares configured for the service class. Possible values are soft and hard.

*Table 480. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics | ACTIVITY METRICS BASE            |

---

## cpu\_shares - WLM dispatcher CPU shares monitor element

The number of WLM dispatcher CPU shares configured for the service class.

*Table 481. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics | ACTIVITY METRICS BASE            |

---

## cpu\_speed - CPU clock speed monitor element

The clock speed of the processors on this host, in MHz.

*Table 482. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_system - Kernel time monitor element

Time spent running kernel code, expressed in processor ticks. Reported for Windows, AIX and Linux systems only. This measurement represents the aggregate for all processors on the system.

Table 483. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information            | ACTIVITY METRICS BASE            |
| ENV_GET_DB2_SYSTEM_RESOURCES table<br>function - Return DB2(r) system information | ACTIVITY METRICS BASE            |

### Usage

- This measurement represents the aggregate for all processors on the system.
- On AIX, this metric is reported for the workload partition (WPAR) and the logical partition (LPAR) on which the DB2 server is running.
- You can use this monitor element along with the related processor timer elements to calculate processor utilization for a specific time interval on the host system. To calculate processor utilization as a percentage, perform the following steps:

1. Use the ENV\_GET\_SYSTEM\_RESOURCES function at the beginning to the time interval to retrieve the values for the following metrics:

- `cpu_usert1` = `cpu_user`
- `cpu_systemt1` = `cpu_system`
- `cpu_idlet1` = `cpu_idle`
- `cpu_waitt1` = `cpu_wait`

2. Repeat the preceding step to determine the timestamps for the same metrics at the end of the time interval for which you want to calculate processor utilization:

- `cpu_usert2` = `cpu_user`
- `cpu_systemt2` = `cpu_system`
- `cpu_idlet2` = `cpu_idle`
- `cpu_iowaitt2` = `cpu_iowait`

3. Calculate processor utilization using the following formula:

$$100 \times \frac{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1})}{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1}) + (\text{cpu\_idle}_{t2} - \text{cpu\_idle}_{t1}) + (\text{cpu\_iowait}_{t2} - \text{cpu\_iowait}_{t1})}$$

---

## cpu\_timebase - Frequency of timebase register increment monitor element

The frequency, in Hz, at which the timebase register is incremented. For Linux and PowerPC® systems only

Table 484. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_total - Number of CPUs monitor element

The number of processors on this host.

Table 485. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

### Usage

The number reported for this monitor element means different things in different operating environments. For example, when returned from a Windows, system, **cpu\_total** refers to the total number of processors installed; on AIX, it represents the number of configured processors.

---

## cpu\_usage\_total - Processor usage monitor element

The overall processor usage on this host including kernel processing time, expressed as a percentage. Reported for AIX, Linux and Windows systems only.

Table 486. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## cpu\_user - Non-kernel processing time monitor element

Time spent running user (non-kernel) code, expressed in processor ticks. Reported for Windows, AIX, and Linux systems only. This measurement represents the aggregate for all processors on the system.

Table 487. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information            | ACTIVITY METRICS BASE            |
| ENV_GET_DB2_SYSTEM_RESOURCES table<br>function - Return DB2(r) system information | ACTIVITY METRICS BASE            |

### Usage

- This measurement represents the aggregate for all processors on the system.
- On AIX, this metric is reported for the workload partition (WPAR) and the logical partition (LPAR) on which the DB2 server is running.
- You can use this monitor element along with the related processor timer elements to calculate processor utilization for a specific time interval on the host system. To calculate processor utilization as a percentage, perform the following steps:
  1. Use the ENV\_GET\_SYSTEM\_RESOURCES function at the beginning to the time interval to retrieve the values for the following metrics:
    - $cpu\_user_{t1} = \mathbf{cpu\_user}$
    - $cpu\_system_{t1} = \mathbf{cpu\_system}$

- `cpu_idlet1` = `cpu_idle`
  - `cpu_waitt1` = `cpu_wait`
2. Repeat the preceding step to determine the timestamps for the same metrics at the end of the time interval for which you want to calculate processor utilization:
    - `cpu_usert2` = `cpu_user`
    - `cpu_systemt2` = `cpu_system`
    - `cpu_idlet2` = `cpu_idle`
    - `cpu_iowaitt2` = `cpu_iowait`
  3. Calculate processor utilization using the following formula:
 
$$100 \times \frac{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1})}{(\text{cpu\_system}_{t2} - \text{cpu\_system}_{t1}) + (\text{cpu\_user}_{t2} - \text{cpu\_user}_{t1}) + (\text{cpu\_idle}_{t2} - \text{cpu\_idle}_{t1}) + (\text{cpu\_iowait}_{t2} - \text{cpu\_iowait}_{t1})}$$

---

## cpu\_utilization - CPU utilization monitor element

The total CPU time consumed by the service class or workload on a particular logical partition divided by the amount of CPU time available on the host or the LPAR in a given period of time.

*Table 488. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics                     | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample workload metrics                               | REQUEST METRICS BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | REQUEST METRICS BASE             |

*Table 489. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document              | REQUEST METRICS BASE |

### Usage

When returned by the `WLM_GET_WORKLOAD_STATS` or the `WLM_GET_SERVICE_SUBCLASS_STATS` function, this monitor element represents the CPU utilization since the last reset of the statistics.

When returned by the `MON_SAMPLE_SERVICE_CLASS_METRICS` or the `MON_SAMPLE_WORKLOAD_METRICS` function, this monitor element represents the CPU utilization since the function was executed.

---

## cpu\_velocity - CPU velocity monitor element

A measure of the amount of contention for the CPU resources, measured on a scale from 0 to 1, with lower numbers meaning greater contention.

CPU velocity is computed by measuring the amount of time that work in a service class has access to the CPU divided by the total time spent accessing the CPU or waiting to access the CPU. It gives a measure of how efficiently the work is being executed relative to how efficiently it could be executed if such work never had to wait for the CPU. The formula is as follows:

$$\text{cpu\_velocity} = \text{total\_cpu\_time} / (\text{total\_cpu\_time} + \text{total\_disp\_run\_queue\_time})$$

The **wlm\_dispatcher** database manager configuration parameter must be set to ON for **cpu\_velocity** to be collected.

*Table 490. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS - Sample service class metrics | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS - Sample workload metrics           | REQUEST METRICS BASE             |

### Usage

The dispatcher is effective at prioritizing a service class or workload when that service class or workload demands more CPU resources at a given instant than can be supplied. In such instances, the work executing in the service class or workload spends time queuing to access the CPU resources. It is when this occurs that the dispatcher can give more of the CPU resources to such a service class or workload by reducing how much of the CPU resources it gives to another. A high CPU velocity indicates that the dispatcher can have little effect on improving response times or throughput for this service class at its current level of CPU demand because this demand is already being met. A low CPU velocity indicates that the dispatcher can potentially have a significant effect on improving response times or throughput for this service class or workload at its current level of CPU demand.

Use this element to determine whether the work executing in a service class or workload is spending a relatively large proportion of its time queuing to use the CPU resources. If the CPU velocity for a service class is low and you want to increase it, you can adjust the WLM dispatcher control of the CPU resources by increasing the number of CPU shares or increasing the CPU limit assigned to the service class that is exhibiting a low CPU velocity.

---

## cputime\_threshold\_id - CPU time threshold ID monitor element

The ID of the CPUTIME threshold that was applied to the activity.

*Table 491. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which CPUTIME threshold, if any, was applied to the activity.

---

### **cputime\_threshold\_value - CPU time threshold value monitor element**

The upper bound of the CPUTIME threshold that was applied to the activity.

*Table 492. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the CPUTIME threshold applied to the activity, if any.

---

### **cputime\_threshold\_violated - CPU time threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CPUTIME threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 493. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the CPUTIME threshold that was applied to the activity.

---

### **cputimeinsec\_threshold\_id - CPU time in service class threshold ID monitor element**

The ID of the CPUTIMEINSEC threshold that was applied to the activity.

*Table 494. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which CPUTIMEINSC threshold, if any, was applied to the activity.

---

### **cputimeinsc\_threshold\_value - CPU time in service class threshold value monitor element**

The upper bound of the CPUTIMEINSC threshold that was applied to the activity.

*Table 495. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the CPUTIMEINSC threshold applied to the activity, if any.

---

### **cputimeinsc\_threshold\_violated - CPU time in service class threshold violated monitor element**

This monitor element returns 'Yes' to indicate that the activity violated the CPUTIMEINSC threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 496. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the CPUTIMEINSC threshold that was applied to the activity.

---

### **create\_nickname - Create Nicknames**

This element contains a count of the total number of times the federated server has created a nickname over an object residing on this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

*Table 497. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to determine the amount of CREATE NICKNAME activity against this data source by this federated server instance or an application. CREATE NICKNAME processing results in multiple queries running against the data source catalogs; therefore, if the value of this element is high, you should determine the cause and perhaps restrict this activity.

---

## create\_nickname\_time - Create Nickname Response Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to process CREATE NICKNAME statements from all applications or a single application running on this federated server instance. The response time is measured since the start of the federated server instance, or the last reset of the database monitor counter, whichever is the latest. The response time is measured as the difference between the time the federated server started retrieving information from the data source to process the CREATE NICKNAME statement, and the time it took to retrieve all the required data from the data source.

Table 498. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine how much actual time was used to create nicknames for this data source.

---

## creator - Application Creator

The authorization ID of the user that pre-compiled the application.

Table 499. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

Table 500. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_dlconn          | -              |
| Statements | event_stmt            | -              |
| Activities | event_activitystmt    | -              |

**Usage** Use this element to help identify the SQL statement that is processing, in conjunction with the CREATOR column of the package section information in the catalogs.

If the CURRENT PACKAGE PATH special register is set, the *creator* value may reflect different values over the lifetime of the SQL statement. If a

snapshot or event monitor record is taken before PACKAGE PATH resolution, the *creator* value will reflect the value flowed in from the client request. If a snapshot or event monitor record is taken after PACKAGE PATH resolution, the *creator* value will reflect the creator of the resolved package. The resolved package will be the package whose *creator* value appears earliest in the CURRENT PACKAGE PATH SPECIAL REGISTER and whose package name and unique ID matches that of the client request.

---

## current\_cf\_gbp\_size - Current cluster caching facility group buffer pool size monitor element

Group buffer pool memory currently in use at the cluster caching facility, in pages with a page size of 4 KB.

*Table 501. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## current\_cf\_lock\_size - Current cluster caching facility lock size monitor element

Global lock memory currently in use, in pages with a page size of 4 KB.

*Table 502. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## current\_cf\_sca\_size - Current cluster caching facility shared communications area size monitor element

Shared communications area memory currently in use, in pages with a page size of 4 KB.

*Table 503. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## current\_cf\_mem\_size - Current cluster caching facility memory size monitor element

Total memory currently in use, in pages with a page size of 4 KB.

*Table 504. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## current\_active\_log - Current Active Log File Number

The file number of the active log file the DB2 database system is currently writing.

Table 505. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 506. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 507. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *first\_active\_log* and *last\_active\_log* elements to determine the range of active log files. Knowing the range of active log files helps you determine the disk space required for log files.

You can also use this element to determine which log files have data to help you identify log files needed for split mirror support.

---

## current\_archive\_log - Current Archive Log File Number

The file number of the log file the DB2 database system is currently archiving. If the DB2 database system is not archiving a log file, the value for this element is `SQLM_LOGFILE_NUM_UNKNOWN`.

Table 508. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 509. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 510. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element to determine if there is a problem archiving log files. Such problems include:

- Slow archive media
- Archive media that is not available

---

## current\_extent - Extent currently being moved monitor element

The numeric identifier of the extent currently being moved by the table space rebalancing process.

Table 511. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_EXTENT_MOVEMENT_STATUS<br>- Get extent movement progress status metrics | ACTIVITY METRICS BASE                        |

---

## current\_request - Current operation request monitor element

The operation currently being processed or most recently processed by the agent.

Table 512. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## cursor\_name - Cursor Name

The name of the cursor corresponding to this SQL statement.

**Element identifier**  
cursor\_name

**Element type**  
information

Table 513. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 514. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks with Details | event_detailed_dlconn | Always collected |
| Statements             | event_stmt            | Always collected |

**Usage** You may use this element to identify the SQL statement that is processing. This name will be used on an OPEN, FETCH, CLOSE, and PREPARE of an SQL SELECT statement. If a cursor is not used, this field will be blank.

---

## data\_object\_pages - Data Object Pages

The number of disk pages consumed by a table. This size represents the base table size only. Space consumed by index objects are reported by *index\_object\_pages*, LOB data is reported by *lob\_object\_pages*, and long data is reported by *long\_object\_pages*.

Table 515. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 516. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

**Usage** This element provides a mechanism for viewing the actual amount of space consumed by a particular table. This element can be used in conjunction with a table event monitor to track the rate of table growth over time.

---

## data\_object\_l\_pages - Table data logical pages monitor element

The number of logical pages used on disk by data contained in this table.

Table 517. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

### Usage

- This value might be less than the amount of space allocated for the object. This can happen when you use the RECLAIM EXTENTS ONLY option with the REORG TABLE command. In this case, reclaimed extents are included in the logical number of pages returned by MON\_GET\_TABLE.
- This value might be less than the amount of space physically allocated for the object. This can happen when you use the REUSE STORAGE option of the TRUNCATE statement. This option causes storage allocated for the table to continue to be allocated, although the storage will be considered empty. In addition, the value for this monitor element might be less than the amount of space logically allocated for the object, because the total space logically allocated includes a small amount of additional meta data.

To retrieve an accurate measure of the logical or physical size of an object, use the ADMIN\_GET\_TAB\_INFO\_V97 function. This function provides more accurate information about the size of objects than you can obtain by multiplying the number of pages reported for this monitor element by the page size.

---

## data\_partition\_id - Data partition identifier monitor element

The identifier of the data partition for which information is returned.

Table 518. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information | ACTIVITY METRICS BASE            |
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details                             | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics   | ACTIVITY METRICS BASE            |

Table 518. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information     | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics                                       | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list  | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list          | ACTIVITY METRICS BASE            |

Table 519. Snapshot monitoring information

| Snapshot level | Logical data grouping | Monitor switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 520. Event monitoring information

| Event type             | Logical data grouping | Monitor switch |
|------------------------|-----------------------|----------------|
| Table                  | event_table           | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |
| Deadlocks              | lock                  | -              |

## Usage

This element is only applicable to partitioned tables and partitioned indexes. Otherwise, the value of this monitor element is NULL.

When returning lock level information, a value of -1 represents a lock which controls access to the whole table.

---

## datasource\_name - Data Source Name

This element contains the name of the data source whose remote access information is being displayed by the federated server. This element corresponds to the 'SERVER' column in SYSCAT.SERVERS.

### Element identifier

datasource\_name

### Element type

information

Table 521. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

**Usage** Use this element to identify the data source whose access information has been collected and is being returned.

---

## **datataginsc\_threshold\_id - Data tag in service class threshold (IN condition) ID**

The ID of the DATATAGINSC IN threshold that was applied to the activity.

Table 522. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### **Usage notes**

- Use this element to understand which DATATAGINSC IN threshold was applied to the activity, if any.

---

## **datataginsc\_threshold\_value - Data tag in service class threshold (IN condition) value**

Comma separated list of data tags in the DATATAGINSC IN threshold that was applied to the activity.

Table 523. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### **Usage notes**

- Use this element to understand the value of the DATATAGINSC IN threshold applied to the activity, if any.

---

## **datataginsc\_threshold\_violated - Data tag in service class threshold (IN condition) violated**

Indicates if the activity has violated the DATATAGINSC IN threshold. Returns 1 if the activity violated the DATATAGINSC IN threshold. Returns 0 if the activity has not violated the threshold.

Table 524. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### Usage notes

- Use this element to determine if the activity violated the DATATAGINSC IN threshold that was applied to the activity.

---

## **datatagnotinsc\_threshold\_id - Data tag in service class threshold (NOT IN condition) ID**

The ID of the DATATAGINSC NOT IN threshold that was applied to the activity.

*Table 525. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### Usage notes

- Use this element to understand which DATATAGINSC NOT IN threshold was applied to the activity, if any.

---

## **datatagnotinsc\_threshold\_value - Data tag in service class threshold (NOT IN condition) value**

Comma separated list of data tags in the DATATAGINSC NOT IN threshold that was applied to the activity.

*Table 526. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### Usage notes

- Use this element to understand the value of the DATATAGINSC NOT IN threshold applied to the activity, if any.

---

## **datatagnotinsc\_threshold\_violated - Data tag in service class threshold (NOT IN condition) violated**

Indicates if the activity has violated the DATATAGINSC NOT IN threshold. Returns 1 if the activity violated the DATATAGINSC NOT IN threshold. Returns 0 if the activity did not violate the threshold.

*Table 527. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE            |

### Usage notes

- Use this element to determine if the activity violated the DATATAGINSC NOT IN threshold that was applied to the activity.

---

## db2\_process\_id - DB2 process ID monitor element

Numeric identifier of the DB2 process running on the reported member.

Table 528. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_DB2_SYSTEM_RESOURCES table function - Return DB2(r) system information | ACTIVITY METRICS BASE            |

---

---

## db2\_process\_name - DB2 process name monitor element

Name of the DB2 process running on the reported member.

Table 529. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_DB2_SYSTEM_RESOURCES table function - Return DB2(r) system information | ACTIVITY METRICS BASE            |

---

---

## db2\_status - Status of DB2 instance monitor element

The current status of the instance of the database manager.

Table 530. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

---

### Usage

You can use this element to determine the state of your database manager instance.

Values for this element are:

| API Constant          | Value | Description   |
|-----------------------|-------|---|
| SQLM_DB2_ACTIVE       | 0     | The database manager instance is active.  |
| SQLM_DB2_QUIESCE_PEND | 1     | The instance and the databases in the instance are in quiesce-pending state. New connections to any instance database are not permitted and new units of work cannot be started. Depending on the quiesce request, active units of work will be allowed to complete or will be rolled back immediately. |
| SQLM_DB2_QUIESCED     | 2     | The instance and the databases in the instance has been quiesced. New connections to any instance database are not permitted and new units of work cannot be started.   |

---

## db2start\_time - Start Database Manager Timestamp

The date and time that the database manager was started using the db2start command.

**Element identifier**  
db2start\_time

**Element type**  
timestamp

Table 531. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

Table 532. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change history | evmonstart            | Always collected |

### Usage

This element can be used with the *time\_stamp* monitor element to calculate the elapsed time since the database manager was started up until the snapshot was taken.

For the change history event monitor, this element can be used to track when deferred database manager configuration parameter updates took effect.

---

## db\_conn\_time - Database activation timestamp monitor element

The date and time of the connection to the database (at the database level, this is the first connection to the database), or when the activate database was issued.

Table 533. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Timestamp              |
| Table Space    | tablespace_list       | Buffer Pool, Timestamp |
| Table          | table_list            | Timestamp              |

Table 534. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Unit of work   | -                     | -                |
| Change history | evmonstart            | Always collected |

### Usage

Use this element with the **disconn\_time** monitor element to calculate the total connection time.

For the change history event monitor, this element can be used to track when deferred database configuration parameter updates took effect.

---

## db\_heap\_top - Maximum Database Heap Allocated

This element is being maintained for DB2 version compatibility. It now measures memory usage, but not exclusively usage by the database heap.

**Note:** The **db\_heap\_top** monitor element is deprecated starting with DB2 Version 9.5. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 535. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 536. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

---

## db\_location - Database Location

The location of the database in relation to the application.

*Table 537. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** Determine the relative location of the database server with respect to the application taking the snapshot. Values are:

- SQLM\_LOCAL
- SQLM\_REMOTE

---

## db\_name - Database name monitor element

The real name of the database for which information is collected or to which the application is connected. This is the name the database was given when created.

*Table 538. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics  | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information                        | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function - Get memory set information                          | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics                     | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample workload metrics                               | ACTIVITY METRICS BASE            |

Table 539. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Database        | dbase                 | Basic          |
| Database        | dbase_remote          | Basic          |
| Application     | appl_id_info          | Basic          |
| Application     | appl_remote           | Basic          |
| Table Space     | tablespace_list       | Buffer Pool    |
| Buffer Pool     | bufferpool            | Buffer Pool    |
| Table           | table_list            | Table          |
| Lock            | db_lock_list          | Basic          |
| Dynamic SQL     | dynsql_list           | Basic          |
| DCS Database    | dcs_dbase             | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 540. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbheader        | Always collected |

## Usage

You may use this element to identify the specific database to which the data applies.

For applications that are not using DB2 Connect to connect to a host or System i<sup>®</sup> database server, you can use this element in conjunction with the **db\_path** monitor element to uniquely identify the database and help relate the different levels of information provided by the monitor.

---

## db\_path - Database Path

The full path of the location where the database is stored on the monitored system.

Table 541. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl_id_info          | Basic          |
| Table Space    | tablespace_list       | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Table          | table_list            | Table          |
| Lock           | db_lock_list          | Basic          |
| Dynamic SQL    | dynsql_list           | Basic          |

Table 542. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbheader        | Always collected |

**Usage** This element can be used with the *db\_name* monitor element to identify the specific database to which the data applies.

---

## db\_status - Status of database monitor element

The current status of the database.

*Table 543. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

### Usage

You can use this element to determine the state of your database.

Values for this field are:

| API Constant           | Value | Description   |
|------------------------|-------|---|
| SQLM_DB_ACTIVE         | 0     | The database is active.   |
| SQLM_DB_QUIESCE_PEND   | 1     | The database is in quiesce-pending state. New connections to the database are not permitted and new units of work cannot be started. Depending on the quiesce request, active units of work will be allowed to complete or will be rolled back immediately. |
| SQLM_DB_QUIESCED       | 2     | The database has been quiesced. New connections to the database are <b>not</b> permitted and new units of work cannot be started.   |
| SQLM_DB_ROLLFWD        | 3     | A rollforward is in progress on the database.   |
| SQLM_DB_ACTIVE_STANDBY | 4     | The database is a read-enabled HADR standby database.   |
| SQLM_DB_STANDBY        | 5     | The database is an HADR standby database.   |

---

## db\_storage\_path - Automatic storage path monitor element

This element shows the full path of a location that is used by the database for placing automatic storage table spaces. There can be 0 or more storage paths associated with a database.

*Table 544. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |

Table 545. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Basic          |

## Usage

Use this element with the **num\_db\_storage\_paths** monitor element to identify the storage paths that are associated with this database.

---

## db\_storage\_path\_id - Storage path identifier

Unique identifier for each occurrence of a storage path in a storage group.

Table 546. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics                     | ACTIVITY METRICS BASE            |

---

## db\_storage\_path\_state - Storage path state monitor element

The automatic storage path state indicates whether the storage path is in use by the database.

Table 547. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |

Table 548. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Basic          |

## Usage

Use this monitor element to determine whether the storage path is in use by the database. The following values are possible:

### NOT\_IN\_USE

There are no table spaces using this storage path on the specified database partition.

### IN\_USE

There are table spaces using this storage path on the specified database partition.

### DROP\_PENDING

This storage path has been dropped, but some table spaces are still using it. Before storage paths are physically dropped from the database, all table

spaces must stop using them. To stop using a dropped storage path, either drop the table space or rebalance the table space using the REBALANCE clause of the ALTER TABLESPACE statement.

## db\_storage\_path\_with\_dpe - Storage path including database partition expression monitor element

Automatic storage path that includes the unevaluated database partition expression.

Table 549. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |

Table 550. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Basic          |

### Usage

Use this monitor element to determine the storage path that was specified for the database as part of the CREATE DATABASE command or the ALTER DATABASE statement, if the storage path contains a database partition expression.

If the storage path does not contain a database partition expression, then this monitor element returns a null value.

## db\_work\_action\_set\_id - Database work action set ID monitor element

If this activity has been categorized into a work class of database scope, this monitor element shows the ID of the work action set associated with the work class set to which the work class belongs. Otherwise, this monitor element shows the value of 0.

Table 551. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| WLM_GET_ACTIVITY_DETAILS_COMPLETE (reported in DETAILS XML document)    | ACTIVITY METRICS BASE                        |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details | ACTIVITY METRICS BASE                        |

Table 552. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

This element can be used with the **db\_work\_class\_id** element to uniquely identify the database work class of the activity, if one exists.

---

## db\_work\_class\_id - Database work class ID monitor element

If this activity has been categorized into a work class of database scope, this monitor element displays the ID of the work class. Otherwise, this monitor element displays the value of 0.

Table 553. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_ACTIVITY_DETAILS_COMPLETE table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 554. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

This element can be used with the **db\_work\_action\_set\_id** element to uniquely identify the database work class of the activity, if one exists.

---

## dbpartitionnum - Database partition number monitor element

In a partitioned database environment, this is the numeric identifier for the database member. For DB2 Enterprise Server Edition and in a DB2 pureScale environment, this value is 0.

Table 555. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO table function - returns compressed index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function - returns index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_MSGS table function - Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure | ACTIVITY METRICS BASE            |
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups  | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries   | ACTIVITY METRICS BASE            |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information                      | ACTIVITY METRICS BASE            |
| AUDIT_ARCHIVE procedure and table function - Archive audit log file   | ACTIVITY METRICS BASE            |

Table 555. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| DBCFCG administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information      | ACTIVITY METRICS BASE            |
| DBPATHS administrative view and ADMIN_LIST_DB_PATHS table function - Retrieve database paths                          | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics  | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS table function - Get rebalance progress for a table space                                    | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics   | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility  | ACTIVITY METRICS BASE            |
| PDLOGMSG_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function - Retrieve problem determination messages | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                                  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                                 | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                               | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUPERCLASS_STATS table function - Return statistics of service superclasses                           | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS table function - Return work action set statistics                                      | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                                   | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics  | ACTIVITY METRICS BASE            |

## Usage

In a DB2 pureScale environment, multiple members operate on a single partition. When running in such a configuration, physical attributes of storage, such as the number of free pages in a table space, are duplicated across all members in the

system. Each member reports the total accurate size for the system. In a multiple partition configuration, the values from each partition must be correlated by the user in order to understand the overall value for the system.

The **dbpartitionnum** monitor element is different to the **data\_partition\_id** monitor element, which is used to identify a data partition that was created by subdividing data in a table based on a value.

---

## dc\_s\_appl\_status - DCS application status monitor element

The status of a DCS application at the DB2 Connect gateway.

*Table 556. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dc_s_appl_info        | Basic          |

### Usage

Use this element for problem determination on DCS applications. Values are:

- **SQLM\_DCS\_CONNECTPEND\_OUTBOUND**  
The application has initiated a database connection from the DB2 Connect gateway to the host database, but the request has not completed yet.
- **SQLM\_DCS\_UOWWAIT\_OUTBOUND**  
The DB2 Connect gateway is waiting for the host database to reply to the application's request.
- **SQLM\_DCS\_UOWWAIT\_INBOUND**  
The connection from the DB2 Connect gateway to the host database has been established and the gateway is waiting for SQL requests from the application. Or the DB2 Connect gateway is waiting on behalf of the unit of work in the application. This usually means that the application's code is being executed.

---

## dc\_s\_db\_name - DCS Database Name

The name of the DCS database as cataloged in the DCS directory.

*Table 557. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Database    | dc_s_dbase            | Basic          |
| DCS Application | dc_s_appl_info        | Basic          |

**Usage** Use this element for problem determination on DCS applications.

---

## ddl\_classification - DDL classification

Classification describing type of DDL executed.

*Table 558. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DDLSTMTEXEC           | Always collected |

## Usage

DDL can be classified as one of:

### DDLSTORAGE

The execution of alter database, buffer pool, partition group, storage group, and table space DDL.

### DDLWLM

The execution of histogram, service class, threshold, work action set, work class set, and workload DDL.

### DDLMONITOR

The execution of event monitor, and usage list DDL.

### DDLSECURITY

The execution of audit policy, grant, mask, permission role, revoke, security label, security label component, security policy, and trusted context DDL.

### DDLSQL

The execution of alias, function, method, module, package, procedure, schema, synonym, transform, trigger, type, variable, and view DDL.

### DDLDATA

The execution of index, sequence, table, and temporary table DDL.

### DDLXML

The execution of XSROBJECT DDL.

### DDLFEDERATED

The execution of nickname/server, type/user mapping, and wrapper DDL.

---

## ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements

This element indicates the number of SQL Data Definition Language (DDL) statements that were executed.

*Table 559. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 560. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to determine the level of database activity at the application or database level. DDL statements are expensive to run due to their impact on the system catalog tables. As a result, if the value of this element is high, you should determine the cause, and possibly restrict this activity from being performed.

You can also use this element to determine the percentage of DDL activity using the following formula:

`ddl_sql_stmts / total number of statements`

This information can be useful for analyzing application activity and throughput. DDL statements can also impact:

- the catalog cache, by invalidating table descriptor information and authorization information that are stored there and causing additional system usage to retrieve the information from the system catalogs
- the package cache, by invalidating sections that are stored there and causing additional processing time due to section recompilation.

Examples of DDL statements are CREATE TABLE, CREATE VIEW, ALTER TABLE, and DROP INDEX.

---

## deadlock\_id - Deadlock Event Identifier

The deadlock identifier for a deadlock.

**Element identifier**

deadlock\_id

**Element type**

information

*Table 561. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch   |
|---------------------------------------|-----------------------|------------------|
| Deadlocks                             | event_deadlock        | Always collected |
| Deadlocks                             | event_dlconn          | Always collected |
| Deadlocks with Details                | event_detailed_dlconn | Always collected |
| Deadlocks with Details History        | event_detailed_dlconn | Always collected |
| Deadlocks with Details History        | event_stmt_history    | Always collected |
| Deadlocks with Details History Values | event_data_value      | Always collected |
| Deadlocks with Details History Values | event_detailed_dlconn | Always collected |
| Deadlocks with Details History Values | event_stmt_history    | Always collected |

**Usage** Use this element in your monitoring application to correlate deadlock connection and statement history event records with deadlock event records.

---

## deadlock\_member - Deadlock member monitor element

The member were the participant is requesting the lock.

*Table 562. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## deadlock\_node - Partition Number Where Deadlock Occurred

Partition number where the deadlock occurred.

**Element identifier**  
deadlock\_node

**Element type**  
information

*Table 563. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_deadlock        | Always collected |
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** This element is relevant only for partitioned databases. Use this in your monitoring application to correlate deadlock connection event records with deadlock event records.

---

## deadlock\_type - Deadlock type monitor element

The type of deadlock that has occurred. The value can be either LOCAL or GLOBAL. In a local deadlock, all participants run on the same member. In a global deadlock, at least one deadlock participant is running on a remote member.

*Table 564. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock                  |                |

---

## deadlocks - Deadlocks detected monitor element

The total number of deadlocks that have occurred.

*Table 565. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |

Table 565. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 566. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Lock           |

For snapshot monitoring, this counter can be reset.

Table 567. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

This element can indicate that applications are experiencing contention problems. These problems could be caused by the following situations:

- Lock escalations are occurring for the database

- An application may be locking tables explicitly when system-generated row locks may be sufficient
- An application may be using an inappropriate isolation level when binding
- Catalog tables are locked for repeatable read
- Applications are getting the same locks in different orders, resulting in deadlock.

You may be able to resolve the problem by determining in which applications (or application processes) the deadlocks are occurring. You may then be able to modify the application to better enable it to run concurrently. Some applications, however, may not be capable of running concurrently.

You can use the connection timestamp monitor elements (**last\_reset**, **db\_conn\_time**, and **appl\_con\_time**) to determine the severity of the deadlocks. For example, 10 deadlocks in 5 minutes is much more severe than 10 deadlocks in 5 hours.

The descriptions for the previously listed related elements may also provide additional tuning suggestions.

---

## deferred - Deferred

Indicates if a change to a configuration parameter value is deferred.

*Table 568. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | DBDBMCFG              | Always collected |

### Usage

The change history event monitor collected this value as:

- Y Change deferred until next database activation
- N Change takes effect immediately

---

## degree\_parallelism - Degree of Parallelism

The degree of parallelism requested when the query was bound.

**Element identifier**  
degree\_parallelism

**Element type**  
information

*Table 569. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

**Usage** Use with `agents_top`, to determine if the query achieved maximum level of parallelism.

---

## del\_keys\_cleaned - Pseudo deleted keys cleaned monitor element

Number of pseudo deleted keys that have been cleaned.

Table 570. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

---

## delete\_sql\_stmts - Deletes

This element contains a count of the total number of times the federated server has issued a DELETE statement to this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

Table 571. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

---

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine the level of database activity directed against this data source by the federated server or an application.

You can also use this element to determine the percentage of write activity against this data source by the federated server or an application, with the following formula:

$$\text{write\_activity} = \frac{(\text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}{(\text{SELECT statements} + \text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}$$

---

## delete\_time - Delete Response Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to DELETES from all applications or a single application running on this federated server instance since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

The response time is measured as the difference in time between the time the federated server submits a DELETE statement to the data source, and the time the data source responds to the federated server, indicating the DELETE has been processed.

Table 572. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

---

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine how much actual time transpires while waiting for DELETES to this data source to be processed. This information can be useful for capacity planning and tuning.

---

## destination\_service\_class\_id – Destination service class ID monitor element

The ID of the service subclass to which an activity was remapped when the threshold violation record to which this element belongs was generated. This element has a value of zero for any threshold action other than REMAP ACTIVITY.

*Table 573. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

### Usage

Use this element to trace the path of an activity through the service classes to which it was remapped. This element can also be used to compute aggregates of how many activities were mapped into a given service subclass.

---

## device\_type - Device type

This element is an identifier for the device type associated with a UTILSTART event.

*Table 574. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILLOCATION          | Always collected |

### Usage

For the change history event monitor, this field determines the interpretation for the LOCATION field:

- A TSM
- C Client
- D Disk
- F Snapshot backup
- L Local
- N Internally generated by DB2
- O Other vendor device support
- P Pipe
- Q Cursor
- R Remove fetch data
- S Server
- T Tape
- U User exit

## diaglog\_write\_wait\_time - Diagnostic log file write wait time monitor element

The time spent waiting on a write to the db2diag log file. The value is given in milliseconds.

Table 575. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 576. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to understand the amount of time spent writing to the db2diag log file. In a partitioned database environment, a high value for this time may indicate contention for the db2diag log file if shared storage is being used for the diagnostic directory path (diagpath). A high value may also indicate excessive logging, for example if **diaglevel** has been set to log all informational messages.

## diaglog\_writes\_total - Total diagnostic log file writes monitor element

The number of times agents have written to the db2diag log file.

Table 577. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 577. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 578. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element with the **diaglog\_write\_wait\_time** monitor element to understand the average amount of time spent writing to the db2diag log file.

## direct\_read\_reqs - Direct read requests monitor element

The number of requests to perform a direct read of one or more sectors of data.

Table 579. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 579. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 580. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 581. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |

Table 581. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activymetrics                                  | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Database      | event_db   | Always collected      |
| Connection    | event_conn   | Always collected      |
| Tablespaces   | event_tablespace                                     | Always collected      |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average number of sectors that are read by a direct read:

$$\text{direct\_reads} / \text{direct\_read\_reqs}$$

## direct\_read\_time - Direct read time monitor element

The elapsed time required to perform the direct reads. This value is given in milliseconds.

Table 582. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |

Table 582. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 583. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 584. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database     | event_db  | Always collected      |

Table 584. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Connection    | event_conn                                 | Always collected      |
| Tablespaces   | event_tablespace                           | Always collected      |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average direct read time per sector:

$\text{direct\_read\_time} / \text{direct\_reads}$

A high average time may indicate an I/O conflict.

---

## direct\_reads - Direct reads from database monitor element

The number of read operations that do not use the buffer pool.

Table 585. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |

Table 585. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                              | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                                | REQUEST METRICS BASE             |

Table 586. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 587. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average number of sectors that are read by a direct read:

$$\text{direct\_reads} / \text{direct\_read\_reqs}$$

When using system monitors to track I/O, this element helps you distinguish database I/O from non-database I/O on the device.

Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:

- Reading LONG VARCHAR columns
- Reading LOB (large object) columns
- Performing a backup

---

## direct\_write\_reqs - Direct write requests monitor element

The number of requests to perform a direct write of one or more sectors of data.

*Table 588. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |

Table 588. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 589. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 590. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average number of sectors that are written by a direct write:

$$\text{direct\_writes} / \text{direct\_write\_reqs}$$

---

## direct\_write\_time - Direct write time monitor element

The elapsed time required to perform the direct writes. This value is reported in milliseconds.

*Table 591. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 591. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 592. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 593. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average direct write time per sector:

$$\text{direct\_write\_time} / \text{direct\_writes}$$

A high average time may indicate an I/O conflict.

---

## direct\_writes - Direct writes to database monitor element

The number of write operations that do not use the buffer pool.

Table 594. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 594. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_WORKLOAD_OCCURRENCE<br>_ACTIVITIES table function - Return a list of activities | REQUEST METRICS BASE             |

Table 595. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 596. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use the following formula to calculate the average number of sectors that are written by a direct write.

$$\text{direct\_writes} / \text{direct\_write\_reqs}$$

When using system monitors to track I/O, this element helps you distinguish database I/O from non-database I/O on the device.

Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:

- Writing LONG VARCHAR columns
- Writing LOB (large object) columns
- Performing a restore
- Performing a load

- Allocating new extents for SMS table space if MPFA is enabled (which is the default)

## disabled\_peds - Disabled partial early distincts monitor element

The number of times that partial early distinct operations were disabled because insufficient sort heap was available.

Table 597. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 598. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 598. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Connection    | event_conn   | -                     |
| Statements    | event_stmt   | -                     |
| Transactions  | event_xact   | -                     |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with the **total\_peds** monitor element to determine if partial early distinct operations are getting sufficient sort heap memory most of the time. If the ratio of the **disabled\_peds** monitor element to the **total\_peds** monitor element is high, your database performance may be sub-optimal. You should consider increasing the sort heap size or the sort heap threshold, or both.

---

## disconn\_time - Database Deactivation Timestamp

The date and time that the application disconnected from the database (at the database level, this is the time the last application disconnected).

### Element identifier

disconn\_time

### Element type

timestamp

Table 599. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Use this element to calculate the elapsed time since:

- The database was active (for information at the database level)
- The connection was active (for information at the connection level).

---

## disconnects - Disconnects

This element contains a count of the total number of times the federated server has disconnected from this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

Table 600. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to determine the total number of times the federated server has disconnected from this data source on behalf of any application. Together with the CONNECT count, this element provides a mechanism by which you can determine the number of applications this instance of the federated server believes is currently connected to a data source.

---

## dl\_conns - Connections involved in deadlock monitor element

The number of connections that are involved in the deadlock.

Table 601. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | event_deadlock        | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Use this element in your monitoring application to identify how many deadlock connection event records will follow in the event monitor data stream.

---

## dynamic\_sql\_stmts - Dynamic SQL Statements Attempted

The number of dynamic SQL statements that were attempted.

Table 602. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 603. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to calculate the total number of successful SQL statements at the database or application level:

```
dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= throughput during monitoring period
```

---

## edu\_ID - Engine dispatchable unit ID monitor element

ID of engine dispatchable unit with which this memory pool is associated.

*Table 604. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_MEMORY_POOL table function - Get memory pool information       | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility | ACTIVITY METRICS BASE            |

### Usage

When returned by the table function MON\_GET\_MEMORY\_POOL, this monitor element is NULL except when the memory pool being described is PRIVATE.

---

## eff\_stmt\_text - Effective statement text monitor element

The effective text of the SQL statement, if the statement was modified as a result of the statement concentrator.

*Table 605. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_ACTIVITY_DETAILS_COMPLETE table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 606. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activity   | event_activitystmt    | Always collected |

### Usage

If the statement concentrator is enabled and if the statement text has been modified as a result of the statement concentrator, then this monitor element contains the effective statement text. Otherwise, this monitor element contains a text string which is 0 bytes long.

---

## effective\_isolation - Effective isolation monitor element

The effective isolation level for this statement.

*Table 607. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |

Table 607. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

Table 608. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Locking       | -                     | -                 |
| Package cache | -                     | COLLECT BASE DATA |

## Usage

Use this element to understand the isolation level that was used during the execution of the statement.

---

## effective\_lock\_timeout - Effective lock timeout monitor element

The effective lock timeout value for this activity. This value is reported in seconds.

Table 609. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

---

## effective\_query\_degree - Effective query degree monitor element

The effective query degree of parallelism for this activity.

Table 610. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 611. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | -                     | -              |

---

## elapsed\_exec\_time - Statement Execution Elapsed Time

At the DCS statement level, this is the elapsed time spent processing an SQL request on a host database server. This value is reported by this server. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type. In contrast to the host\_response\_time element, this element does not include the network elapsed time between DB2 Connect and the host database server. At other levels, this value represents the sum of the host

execution times for all the statements that were executed for a particular database or application, or for those statements that used a given number of data transmissions.

Table 612. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| Database          | dbase                 | Statement, Timestamp |
| Application       | appl                  | Statement, Timestamp |
| DCS Database      | dc_s_dbase            | Statement, Timestamp |
| DCS Application   | dc_s_appl             | Statement, Timestamp |
| DCS Statement     | dc_s_stmt             | Statement, Timestamp |
| Data Transmission | stmt_transmissions    | Statement, Timestamp |

For snapshot monitoring at the statement level, this counter cannot be reset. This counter can be reset at other levels.

**Usage** Use this element, along with other elapsed time monitor elements, to evaluate the database server's processing of SQL requests and to help isolate performance issues.

Subtract this element from the `host_response_time` element to calculate the network elapsed time between DB2 Connect and the host database server.

**Note:** For the `dc_s_dbase`, `dc_s_appl`, `dc_s_stmt` and `stmt_transmissions` levels, the `elapsed_exec_time` element applies only to z/OS® databases. If the DB2 Connect gateway is connecting to a Windows, Linux, AIX, or other UNIX database, the `elapsed_exec_time` is reported as zero.

---

## empty\_pages\_deleted - Empty pages deleted monitor element

The number of pseudo empty pages that have been deleted. Pseudo empty pages are pages where all the keys have been pseudo deleted.

Table 613. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## empty\_pages\_reused - Empty pages reused monitor element

The number of pseudo empty pages that have been reused. Pseudo empty pages are pages where all the keys have been pseudo deleted.

Table 614. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## entry\_time - Entry time monitor element

The time at which this activity entered the system.

*Table 615. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES table function - Return a list of activities                       | ACTIVITY METRICS BASE            |

### Usage

---

## estimated\_cpu\_entitlement - Estimated CPU entitlement monitor element

The percentage of total CPU on a host or an LPAR that a service subclass is configured to consume based on its CPU shares assuming that it consumes no more and no less than what it is configured to consume. The determination of which service classes participate in its calculation is based on the actual CPU utilization measured over the sampling period versus the WLM\_DISP\_MIN\_UTIL database manager configuration setting. The impact of a CPU limit on a service class itself, on the service classes with which it competes, or on a parent service class (if it has one) are not taken into account in the calculation.

*Table 616. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics | ACTIVITY METRICS BASE            |

---

## estimatedsqlcost\_threshold\_id - Estimated SQL cost threshold ID monitor element

The ID of the ESTIMATEDSQLCOST threshold that was applied to the activity.

*Table 617. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand which ESTIMATEDSQLCOST threshold, if any, was applied to the activity.

---

## estimatedsqlcost\_threshold\_value - Estimated SQL cost threshold value monitor element

The upper bound of the ESTIMATEDSQLCOST threshold that was applied to the activity.

*Table 618. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to understand the value of the ESTIMATEDSQLCOST threshold applied to the activity, if any.

---

## estimatedsqlcost\_threshold\_violated - Estimated SQL cost threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the ESTIMATEDSQLCOST threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 619. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

Use this element to determine if the activity violated the ESTIMATEDSQLCOST threshold that was applied to the activity.

---

## event\_id - Event ID monitor element

An identifier associated with an event that is used in conjunction with other monitor elements to uniquely identify the event. This monitor element forms a part of the **xmlid** monitor element in the interfaces that return that element.

Table 620. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Element Collection Level |
|----------------|--|----------------------------------|
| Change History | changesummary<br>dbdbmcfg<br>ddlstmexec<br>evmonstart<br>regvar<br>txncompletion<br>utillocation<br>utilphase<br>utilstart<br>utilstop   | Always collected                 |
| Locking        | <ul style="list-style-type: none"><li>• event_lock</li><li>• event_lock_participants</li><li>• event_lock_participants_activities</li><li>• event_lock_activity_values</li></ul> | Always collected                 |
| Package Cache  | <ul style="list-style-type: none"><li>• event_pkgcache</li><li>• event_pkgcache_metrics</li><li>• event_pkgcache_stmt_args</li></ul>   | Always collected                 |
| Unit of Work   | event_uow  | Always collected                 |

### Usage

The value for this identifier is different depending on the type of event record in which it appears:

#### Locking event monitor records

An numeric identifier for the event. The ID is recycled at database activation time. Uniqueness is guaranteed by the combination of **event\_timestamp**, **event\_id**, **member**, and **event-type**.

#### Unit of work event monitor records

An alias of the UOW ID that is unique per connection. Uniqueness is guaranteed by the combination of **event\_timestamp**, **event\_id**, **event-type**, **member**, and **appl\_id**.

#### Package cache event monitor record

An numeric identifier for the event. The ID is recycled at database activation time. Uniqueness is guaranteed by the combination of **event\_timestamp**, **event\_id**, **member**, and **event-type**.

#### Change history event monitor record

An numeric identifier for the event. The ID is recycled at database activation time. The uniqueness of events is guaranteed by the combination of **event\_timestamp**, **event\_id**, **member**, and **event-type**.

---

## event\_monitor\_name - Event Monitor Name

The name of the event monitor that created the event data stream.

### Element identifier

event\_monitor\_name

### Element type

information

Table 621. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |

**Usage** This element allows you to correlate the data that you are analyzing to a specific event monitor in the system catalog tables. This is the same name that can be found in the NAME column of the SYSCAT.EVENTMONITORS catalog table, which is the name specified on the CREATE EVENT MONITOR and SET EVENT MONITOR statements.

---

## event\_time - Event Time

The date and time an event occurred.

Table 622. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Tablespaces | event_tablespace      | -              |
| Tables      | event_table           | -              |

**Usage** You can use this element to help relate events chronologically.

---

## event\_timestamp - Event timestamp monitor element

The time that this event record was generated by the database manager.

Table 623. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Element Collection Level |
|----------------|--|----------------------------------|
| Change History | changesummary<br>dbdbmcfgr<br>ddlstmtexec<br>evmonstart<br>regvar<br>txncompletion<br>utillocation<br>utilphase<br>utilstart<br>utilstop                                 | Always collected                 |
| Locking        | <ul style="list-style-type: none"><li>event_lock</li><li>event_lock_participants</li><li>event_lock_participants_activities</li><li>event_lock_activity_values</li></ul> | Always collected                 |

Table 623. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping  | Monitor Element Collection Level |
|---------------|--|----------------------------------|
| Package Cache | <ul style="list-style-type: none"> <li>event_pkgcache</li> <li>event_pkgcache_metrics</li> <li>event_pkgcache_stmt_args</li> </ul> | Always collected                 |
| Unit of Work  | <ul style="list-style-type: none"> <li>event_uow</li> </ul>  | Always collected                 |

## event\_type - Event Type monitor element

The event type of the event that is being reported. Reported by the change history, locking, and package cache event monitors only.

Many even monitors capture only one type of event. For example, a unit of work event monitor records a unit of work completion event that captures data related to the unit of work when it is completed. Other event monitors, such as the database, table, buffer pool or table space event monitors each capture one type of event after database deactivation.

Some event monitors, however, generate different types of events. These event monitors report the type of event in the **event\_type** monitor element. Table 624 shows the event monitors that return this monitor element. Table 625 on page 813 shows the possible values that this monitor element can have.

Table 624. Event Monitoring Information

| Event Monitor  | Logical Data Grouping   | Monitor Element Collection Level |
|----------------|---|----------------------------------|
| Change History | changesummary<br>dbdbmcfg<br>ddlstmtexec<br>evmonstart<br>regvar<br>txncompletion<br>utillocation<br>utilphase<br>utilstart<br>utilstop                                       | Always collected                 |
| Locking        | <ul style="list-style-type: none"> <li>event_lock</li> <li>event_lock_participants</li> <li>event_lock_participants_activities</li> <li>event_lock_activity_values</li> </ul> | Always collected                 |
| Unit of work   | Not applicable. For more information, see the usage notes for this event monitor in Table 625 on page 813.  | Always collected                 |
| Package cache  | <ul style="list-style-type: none"> <li>pkgcache</li> </ul>  | Always collected                 |

Table 625. Possible values for event\_type

| Event monitor  | Possible values for event_type element  | Usage notes  |
|----------------|---|--|
| Change history | <ul style="list-style-type: none"> <li>• DBCFG</li> <li>• DBCFGVALUES</li> <li>• DBMCFG</li> <li>• DBMCFGVALUES</li> <li>• REGVAR</li> <li>• REGVARVALUES</li> <li>• DDLSTMTEXEC</li> <li>• TXNCOMPLETION</li> <li>• EVMONSTART</li> <li>• UTILSTART</li> <li>• UTILSTOP</li> <li>• UTILSTARTPROC</li> <li>• UTILSTOPPROC</li> <li>• UTILPHASESTART</li> <li>• UTILPHASESTOP</li> </ul> | The <b>event_type</b> monitor element is included as a column in the output of this event monitor.   |
| Locking        | <ul style="list-style-type: none"> <li>• LOCKTIMEOUT</li> <li>• LOCKWAIT</li> <li>• DEADLOCK</li> </ul>   | The <b>event_type</b> monitor element is included as a column in the output of this event monitor when it writes to either regular or unformatted event (UE) tables. It is also included in the output created by the routines EVMON_FORMAT_UE_TO_TABLES or EVMON_FORMAT_UE_TO_XML.  |
| Package cache  | <ul style="list-style-type: none"> <li>• PKGCACHEBASE</li> <li>• PKGCACHEDETAILED</li> </ul>  | The <b>event_type</b> monitor element is included as a column in the output of this event monitor when it writes to unformatted event (UE) tables, but not regular tables. It is also included in the output created by the routines EVMON_FORMAT_UE_TO_TABLES or EVMON_FORMAT_UE_TO_XML.  |
| Unit of work   | <ul style="list-style-type: none"> <li>• UOW</li> </ul>   | The <b>event_type</b> monitor element is included as a column in the output of this event monitor only when it writes to an unformatted event (UE) table. Moreover, it is included only in the UE table created by this event monitor; it is not included in the output created by the routines EVMON_FORMAT_UE_TO_TABLES or EVMON_FORMAT_UE_TO_XML. |

## evmon\_activates - Number of event monitor activations

The number of times an event monitor has been activated.

### Element identifier

evmon\_activates

### Element type

counter

This monitor element is not reported by all event monitors. See Table 626 on page 814 for more details.

Table 626. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Database               | event_db              | Always collected |
| Tables                 | event_table           | Always collected |
| Tablespaces            | event_tablespace      | Always collected |
| Bufferpools            | event_bufferpool      | Always collected |
| Deadlocks              | event_deadlock        | Always collected |
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |
| Statements             | event_stmt_history    | Always collected |
| Connections*           | -                     |                  |
| Transactions*          | -                     |                  |
| Statistics*            | -                     |                  |
| Threshold violations*  | -                     |                  |

\* This event monitor updates the `evmon_activates` column only in the catalog table `SYSCAT.EVENTMONITORS`. The monitor element is not included in any of the logical data groups that are written to event monitor output tables.

**Usage** Use this element to correlate information returned by the event monitors listed in Table 626. This element is applicable only to write-to-table event monitors; it is not maintained for event monitors that write to a file or pipe.

All of the event monitors in Table 626 update the `evmon_activates` column of the `SYSCAT.EVENTMONITORS` catalog table when the event monitor is activated. This change is logged, so the `DATABASE CONFIGURATION` displays:

```
All committed transactions have been written to disk = NO
```

If an event monitor is created with the `AUTOSTART` option, and the first user `CONNECTS` to the database and immediately `DISCONNECTS` so that the database is deactivated, a log file is produced.

Unless otherwise noted, these event monitors also include the value for `evmon_activates` as a column in the tables to which event data is written.

Event monitors not included in this table do not report the `evmon_activates` monitor element.

---

## evmon\_wait\_time - Event monitor wait time monitor element

The amount of time that an agent waited for an event monitor record to become available.

A wait occurs when the agent tries to write an event monitor record and the agent is blocked until a fast writer record becomes available. Fast writers are used for high volume, parallel writing of event monitor data to tables, files, or pipes.

The unit of measurement is milliseconds.

Table 627. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                                    | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details   | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics   | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                       | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - Get detailed metrics for package cache entries (reported in the DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in the DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics   | REQUEST METRICS BASE   |

Table 628. Event Monitoring Information

| Event Type   | Event monitor output type                                  | Logical Data Grouping                                    | Monitor Switch       |
|--------------|--|--|----------------------|
| Unit of work | TABLE, FILE, PIPE  | uow_metrics  | REQUEST METRICS BASE |
| Unit of work | UE TABLE, processed by the EVMON_FORMAT_UE_TO_XML function | Reported in the system_metrics document Always collected | REQUEST METRICS BASE |

Table 628. Event Monitoring Information (continued)

| Event Type    | Event monitor output type                                     | Logical Data Grouping   | Monitor Switch        |
|---------------|---|---|-----------------------|
| Unit of work  | UE TABLE, processed by the EVMON_FORMAT_UE_TO_TABLES function | Reported in the metrics document of the UOW_EVENT table and in the UOW_METRICS table Always collected           | REQUEST METRICS BASE  |
| Package cache | TABLE, FILE, PIPE   | pkgcache_metrics  | ACTIVITY METRICS BASE |
| Package cache | UE TABLE, processed by the EVMON_FORMAT_UE_TO_XML function    | Reported in the activity_metrics document Always collected  | ACTIVITY METRICS BASE |
| Package cache | UE TABLE, processed by the EVMON_FORMAT_UE_TO_TABLES function | Reported in the metrics document of the PKGCACHE_EVENT table and in the PKGCACHE_METRICS table Always collected | ACTIVITY METRICS BASE |
| Activities    | TABLE, FILE, PIPE   | event_activymetrics   | ACTIVITY METRICS BASE |
| Activities    | TABLE, FILE, PIPE   | event_activity (reported in the DETAILS XML document)   | ACTIVITY METRICS BASE |
| Statistics    | TABLE, FILE, PIPE   | event_scstats (reported in the DETAILS XML document)  | REQUEST METRICS BASE  |
| Statistics    | TABLE, FILE, PIPE   | event_wlstats (reported in the DETAILS XML document)  | REQUEST METRICS BASE  |

## evmon\_waits\_total - Event monitor total waits monitor element

The number of times that an agent waited for an event monitor record to become available.

A wait occurs when the agent tries to write an event monitor record and the agent is blocked until a fast writer record becomes available. Fast writers are used for high volume, parallel writing of event monitor data to tables, files, or pipes.

Table 629. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics                     | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics    | REQUEST METRICS BASE   |

Table 629. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - Get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in the DETAILS XML document)            | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics   | REQUEST METRICS BASE             |

Table 630. Event Monitoring Information

| Event Type    | Event monitor output type                                     | Logical Data Grouping   | Monitor Switch        |
|---------------|---|---|-----------------------|
| Unit of work  | TABLE   | uow_metrics   | REQUEST METRICS BASE  |
| Unit of work  | UE TABLE, processed by the EVMON_FORMAT_UE_TO_XML function    | Reported in the system_metrics document Always collected  | REQUEST METRICS BASE  |
| Unit of work  | UE TABLE, processed by the EVMON_FORMAT_UE_TO_TABLES function | Reported in the metrics document of the UOW_EVENT table and in the UOW_METRICS table Always collected           | REQUEST METRICS BASE  |
| Package cache | TABLE   | pkgcache_metrics  | ACTIVITY METRICS BASE |
| Package cache | UE TABLE, processed by the EVMON_FORMAT_UE_TO_XML function    | Reported in the activity_metrics document Always collected  | ACTIVITY METRICS BASE |
| Package cache | UE TABLE, processed by the EVMON_FORMAT_UE_TO_TABLES function | Reported in the metrics document of the PKGCACHE_EVENT table and in the PKGCACHE_METRICS table Always collected | ACTIVITY METRICS BASE |

Table 630. Event Monitoring Information (continued)

| Event Type | Event monitor output type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---------------------------|---|-----------------------|
| Activities | TABLE                     | event_activitiymetrics                                | ACTIVITY METRICS BASE |
| Activities | TABLE                     | event_activity (reported in the DETAILS_XML document) | ACTIVITY METRICS BASE |
| Statistics | TABLE                     | event_scstats (reported in the DETAILS_XML document)  | REQUEST METRICS BASE  |
| Statistics | TABLE                     | event_wlstats (reported in the DETAILS_XML document)  | REQUEST METRICS BASE  |

## executable\_id - Executable ID monitor element

An opaque binary token generated on the data server that uniquely identifies the SQL statement section that was executed. For non-SQL activities, a 0-length string value is returned.

Table 631. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list                     | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                      | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                       | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE            |

Table 632. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Activity      | event_activitiystmt   | -                 |
| Package cache | -                     | COLLECT BASE DATA |

## Usage

Use this monitor element as input to different monitoring interfaces to obtain data about the section. The MON\_GET\_PKG\_CACHE\_STMT table function, which is used to get SQL statement activity metrics in the package cache, takes the executable ID as input.

---

### executable\_list\_size - Size of executable list monitor element

The number of entries that are present within the executable ID listing for a particular unit of work.

Table 633. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Unit of Work | uow                   |                |

---

### executable\_list\_truncated - Executable list truncated monitor element

Indicates whether the executable list is truncated. Possible values are YES or NO. The list can be truncated if there is insufficient memory available to store the entire executable list during processing.

Table 634. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Unit of Work | uow                   |                |

---

### evmon\_flushes - Number of Event Monitor Flushes

The number of times the FLUSH EVENT MONITOR SQL statement has been issued.

**Element identifier**

evmon\_flushes

**Element type**

information

Table 635. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tables      | event_table           | Always collected |
| Tablespaces | event_tablespace      | Always collected |
| Bufferpools | event_bufferpool      | Always collected |

**Usage** This identifier increments with each successive FLUSH EVENT MONITOR SQL request processed by the database manager after an application has connected to the database. This element helps to uniquely identify database, table, table space and buffer pool data.

---

## executable\_id - Executable ID monitor element

An opaque binary token generated on the data server that uniquely identifies the SQL statement section that was executed. For non-SQL activities, a 0-length string value is returned.

*Table 636. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list                     | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                      | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                       | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE            |

*Table 637. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Activity      | event_activystmt      | -                 |
| Package cache | -                     | COLLECT BASE DATA |

### Usage

Use this monitor element as input to different monitoring interfaces to obtain data about the section. The MON\_GET\_PKG\_CACHE\_STMT table function, which is used to get SQL statement activity metrics in the package cache, takes the executable ID as input.

---

## execution\_id - User Login ID

The ID that the user specified when logging in to the operating system. This ID is distinct from auth\_id, which the user specifies when connecting to the database.

*Table 638. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Application    | appl                  | Basic          |

Table 638. Snapshot Monitoring Information (continued)

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcx_appl_info         | Basic          |

Table 639. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**Usage** You can use this element to determine the operating system userid of the individual running the application that you are monitoring.

---

## failed\_sql\_stmts - Failed Statement Operations

The number of SQL statements that were attempted, but failed.

Table 640. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Database        | dbase                 | Basic          |
| Database        | dbase_remote          | Basic          |
| Application     | appl                  | Basic          |
| Application     | appl_remote           | Basic          |
| DCS Database    | dcx_dbase             | Basic          |
| DCS Application | dcx_appl              | Basic          |

For snapshot monitoring, this counter can be reset.

Table 641. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to calculate the total number of successful SQL statements at the database or application level:

```

dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= throughput during monitoring period

```

This count includes all SQL statements that received a negative SQLCODE.

This element may also help you in determining reasons for poor performance, since failed statements mean time wasted by the database manager and as a result, lower throughput for the database.

---

## fcm\_congested\_sends - FCM congested sends monitor element

The number of send operations to a remote member that were delayed due to congestion.

A state of congestion occurs when data sent on a network connection are blocked because previously sent data not being received at the remote member. Congestion can be the result latency in processing by the receiver, or network issues that cause packet loss and retransmission of data.

*Table 642. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

---

## **fcm\_congestion\_time - FCM congestion time monitor element**

The duration of time that the FCM network connection to a remote member was in a state of congestion. Unit of measure is milliseconds.

A state of congestion occurs when data sent on a network connection are blocked because previously sent data not being received at the remote member. Congestion can be the result latency in processing by the receiver, or network issues that cause packet loss and retransmission of data.

*Table 643. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

---

## **fcm\_num\_congestion\_timeouts - FCM congestion timeouts monitor element**

The number of times that a state of congestion on an FCM network connection did not resolve itself within the allowable time period. A timeout occurs when the sender drops the connection because the remote member was temporarily unreachable.

A state of congestion occurs when data sent on a network connection are blocked because previously sent data not being received at the remote member. Congestion can be the result latency in processing by the receiver, or network issues that cause packet loss and retransmission of data.

*Table 644. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

---

## **fcm\_num\_conn\_lost - FCM lost connections monitor element**

The number of times that an FCM network connection to a remote member was unexpectedly disconnected.

*Table 645. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

---

## fcf\_num\_conn\_timeouts - FCF connection timeouts monitor element

The number of times out out when attempting to establish an FCF network connection with a remote member.

Table 646. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCF_CONNECTION_LIST - Get details for all FCF connections | ACTIVITY METRICS BASE            |

---

## fcf\_message\_rcv\_volume - FCF message received volume monitor element

The amount of data received for internal requests (such as RPCs) distributed by the FCF communications layer. This value is reported in bytes.

Table 647. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_FCF_CONNECTION_LIST - Get details for all FCF connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 647. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 648. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine how much of the data volume being sent through the FCM subsystem is for request or reply message traffic as opposed to actual table data.

## fcm\_message\_recv\_wait\_time - FCM message received wait time monitor element

The time spent by an agent waiting for an FCM reply message containing the results of a previously sent FCM request message. This value reflects both the time required to send the response between partitions using FCM and the time required for the subagent to process the request message. The value is given in milliseconds.

Table 649. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                           | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 649. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 650. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

This element can be used to determine how much time was spent on a given partition waiting for requests to be processed on other partitions in a multi-partition instance.

## fcmessage\_recvs\_total - Total FCM message receives monitor element

The total number of buffers received as part of an FCM reply message containing the results of a previously sent FCM request message.

Table 651. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 652. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |

Table 652. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## Usage

This element can be used to determine both the average volume per FCM message received, as well as the average time spent waiting for a single FCM message to be received.

## fcm\_message\_send\_volume - FCM message send volume monitor element

Amount of data volume sent via internal FCM requests. This value is reported in bytes.

Table 653. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 653. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 654. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

Use this element to determine how much of the data volume being sent through the FCM subsystem is used for sending request and reply message traffic, as opposed to sending actual table data.

## fcmessage\_send\_wait\_time - FCM message send wait time monitor element

The time spent blocking on an FCM message send. The value is given in milliseconds. This monitor element reflects the time spent blocking for FCM buffers to be flushed from an FCM channel when distributing internal requests on the database system.

Table 655. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW -<br>Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 655. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 656. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

Use this element to determine how much time agents are spending waiting to send an FCM request message through the FCM subsystem. Depending on how busy the FCM daemons are, an agent may need to wait when attempting to send messages.

---

### **fcm\_message\_sends\_total - Total FCM message sends monitor element**

The total number of buffers distributed as part of internal requests using the FCM communications mechanism.

*Table 657. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b>  |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 658. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine both the average amount of data sent per FCM request message, as well as the average amount of time waited per FCM message.

## fcm\_rcv\_volume - FCM received volume monitor element

The total amount of data received via the FCM communications layer. This value is reported in bytes.

Table 659. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries  | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 659. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 660. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Indicates the total volume of data received on this partition using FCM, including both message traffic and table queue data.

---

## fcm\_rcv\_wait\_time - FCM received wait time monitor element

The total time spent waiting to receive data through FCM. The value is given in milliseconds.

Table 661. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 661. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element  | Collection Level |
|--|------------------|------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS | BASE             |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS  | BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS  | BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS  | BASE             |

Table 662. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine the total time spent waiting to receive data through FCM on this database partition. This includes both data from replies to request messages as well as table queue data.

---

### **fcm\_recvs\_total - FCM receives total monitor element**

Total number of buffers received for internal requests using the FCM communications mechanism. The `fcm_recvs_total` monitor element value is the sum of the values for the `fcm_message_recvs_total` and `fcm_tq_recvs_total` monitor elements.

*Table 663. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b>  |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 664. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element together with the **fcm\_recv\_wait\_time** monitor element to determine the average wait time per FCM receive operation as well as the average volume returned from an FCM receive operation.

## fcm\_send\_volume - FCM send volume monitor element

The total amount of data distributed by the FCM communications layer. This value is reported in bytes.

Table 665. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 665. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 666. Event Monitoring Information

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activitymetrics                      | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the total volume of data sent using FCM, including both message traffic and table queue data.

## fcm\_send\_wait\_time - FCM send wait time monitor element

The time spent blocking on an FCM send operation. This includes time spent waiting for buffers for internal requests to be flushed and time spent waiting for window count acknowledgements when sending data over table queues. The value is given in milliseconds.

Table 667. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                           | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 667. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 668. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine the total time spent waiting to send data through FCM. This includes both request messages and table queue data.

## fcm\_sends\_total - FCM sends total monitor element

The total number of buffers sent using the internal FCM communications layer.

Table 669. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 670. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 670. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

Use this element to determine the average wait time per FCM receive operation as well as the average volume returned from an FCM receive operation.

## fcm\_tq\_recv\_volume - FCM table queue received volume monitor element

The amount of data received on table queues by the FCM communications layer. This value is reported in bytes.

Table 671. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 671. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 672. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the total volume of data received through table queues.

## fcm\_tq\_recv\_wait\_time - FCM table queue received wait time monitor element

The time spent waiting to receive the next buffer from a table queue. The value is given in milliseconds.

Table 673. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                           | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 673. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 674. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine how much time agents are spending waiting to receive data on table queues.

## fcm\_tq\_recvs\_total - FCM table queue receives total monitor element

The total number of buffers received from table queues using the internal FCM communications mechanism.

Table 675. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 676. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 676. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with **fcm\_tq\_recv\_volume** and **fcm\_tq\_recv\_wait\_time** to determine the average wait time and volume per table queue buffer received.

## fcm\_tq\_send\_volume - FCM table queue send volume monitor element

The amount of data sent over table queues by the FCM communications layer. This value is reported in bytes.

Table 677. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 677. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 678. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine the total volume of data sent over FCM through table queue buffer sends.

## fcm\_tq\_send\_wait\_time - FCM table queue send wait time monitor element

The time spent waiting to send the next buffer through a table queue. This reflects the time spent waiting for window count acknowledgements from the receiver end of the table queue. The value is given in milliseconds.

Table 679. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                           | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 679. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 680. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this monitor element to determine how much time is being spent waiting to send a data buffer over FCM through a table queue.

## fcm\_tq\_sends\_total - FCM table queue send total monitor element

The total number of buffers containing table queue data sent using the internal FCM communications mechanism.

Table 681. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 682. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 682. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with **fcm\_tq\_send\_volume** and **fcm\_tq\_send\_wait\_time** monitor elements to determine the average data volume and time waited for buffer sent using a table queue.

## fetch\_count - Number of Successful Fetches

The number of successful physical fetches or the number of attempted physical fetches, depending on the snapshot monitoring level.

- For the stmt and dynsql snapshot monitoring levels and the statement event type: the number of successful fetches performed on a specific cursor.
- For the dcs\_stmt snapshot monitoring level: The number of attempted physical fetches during a statement's execution (regardless of how many rows were fetched by the application). In this situation, **fetch\_count** represents the number of times the server needed to send a reply data back to the gateway while processing a statement.

Table 683. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

For Dynamic SQL snapshot monitoring, this counter can be reset.

Table 684. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statements | event_stmt            | Always collected |

## Usage

You can use this element to gain insight into the current level of activity within the database manager.

For performance reasons, a statement event monitor does not generate a statement event record for every FETCH statement. A record event is only generated when a FETCH returns a non-zero SQLCODE.

---

## files\_closed - Database files closed monitor element

The total number of database files closed.

*Table 685. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE                     |

*Table 686. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 687. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

### Usage

The database manager opens files for reading and writing into and out of the buffer pool. The maximum number of database files open by an application at any time is controlled by the **maxfilop** configuration parameter. If the maximum is reached, one file will be closed before the new file is opened. Note that the actual number of files opened may not equal the number of files closed.

You can use this element to help you determine the best value for the **maxfilop** configuration parameter.

---

## first\_active\_log - First Active Log File Number

The file number of the first active log file.

### Element identifier

first\_active\_log

### Element type

information

*Table 688. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

Table 689. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 690. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *last\_active\_log* and *current\_active\_log* elements to determine the range of active log files. Knowing the range of active log files helps you determine the disk space required for log files.

You can also use this element to determine which log files have data to help you identify log files needed for split mirror support.

## first\_overflow\_time - Time of First Event Overflow

The date and time of the first overflow recorded by this overflow record.

Table 691. Event Monitoring Information

| Event Type      | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Overflow Record | event_overflow        | -              |

**Usage** Use this element with *last\_overflow\_time* to calculate the elapsed time for which the overflow record was generated.

## fs\_caching - File system caching monitor element

Indicates whether a particular table space uses file system caching. If **fs\_caching** is 0, file system caching is enabled. If **fs\_caching** is 1, file system caching is disabled.

Table 692. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

Table 693. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table space    | tablespace            | Basic          |

Table 694. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Tablespaces | event_tablespace      | -              |

---

## fs\_id - Unique file system identification number monitor element

This element shows the unique identification number provided by the operating system for a file system pointed to by a storage path or container.

*Table 695. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics                     | ACTIVITY METRICS BASE            |

*Table 696. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Buffer Pool    |

### Usage

Use this element together with the following elements to gather data on space utilization for the database:

- **db\_storage\_path**
- **sto\_path\_free\_sz**
- **fs\_used\_size**
- **fs\_total\_size**

---

## fs\_total\_size - Total size of a file system monitor element

This element shows the capacity (in bytes) of a file system pointed to by a storage path or container.

*Table 697. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics                     | ACTIVITY METRICS BASE            |

*Table 698. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Buffer Pool    |

### Usage

You can use this element together with the following elements to gather data on space utilization for the database:

- **db\_storage\_path**
- **sto\_path\_free\_sz**

- **fs\_used\_size**
- **fs\_id**

---

## **fs\_used\_size - Amount of space used on a file system monitor element**

This element shows the amount of space (in bytes) already used on a file system pointed to by a storage path or container.

*Table 699. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics                     | ACTIVITY METRICS BASE            |

*Table 700. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Buffer Pool    |

### **Usage**

You can use this element together with the following elements to gather data on space utilization for the database:

- **db\_storage\_path**
- **sto\_path\_free\_sz**
- **fs\_total\_size**
- **fs\_id**

---

## **global\_transaction\_id - Global transaction identifier monitor element**

The XA transaction ID in use at the time the event occurred.

*Table 701. Event Monitoring Information*

| Event Type     | Logical Data Grouping        | Monitor Switch   |
|----------------|------------------------------|------------------|
| Change history | ddlstmtexec<br>txncompletion | Always collected |
| Unit of Work   | uow                          | Always collected |

### **Usage**

For the change history event monitor, this is the global transaction ID in use at the time the event occurred. This is the data field in the SQLP\_GXID structure that is part of the transaction logs.

---

## gw\_comm\_error\_time - Communication Error Time

The date and time when the most recent communication error (SQL30081) occurred while a DCS application was attempting to connect to a host database, or while it was processing an SQL statement.

*Table 702. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Timestamp      |

### Usage

Use this element for problem determination, in conjunction with Communication Error and the communication error logged in administration notification log.

---

## gw\_comm\_errors - Communication Errors

The number of times that a communication error (SQL30081) occurred while a DCS application was attempting to connect to a host database, or while it was processing an SQL statement.

*Table 703. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** By monitoring the number of communication errors over time, you can assess whether your DB2 Connect gateway has connectivity problems with a particular host database. You can establish what you consider to be a normal error threshold, so that any time the number of errors exceeds this threshold an investigation of the communication errors should be made.

Use this element for problem determination, in conjunction with the communication error logged in administration notification log.

---

## gw\_con\_time - DB2 Connect Gateway First Connect Initiated

The date and time when the first connection to the host database was initiated from the DB2 Connect gateway.

*Table 704. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Database    | dcs_dbase             | Timestamp      |
| DCS Application | dcs_appl              | Timestamp      |

**Usage** Use this element for problem determination on DCS applications.

---

## gw\_connections\_top - Maximum Number of Concurrent Connections to Host Database

The maximum number of concurrent connections to a host database that have been handled by the DB2 Connect gateway since the first database connection.

**Element identifier**  
gw\_connections\_top

**Element type**  
watermark

*Table 705. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcс_dbase             | Basic          |

**Usage** This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

---

## gw\_cons\_wait\_client - Number of Connections Waiting for the Client to Send Request

The current number of connections to host databases being handled by the DB2 Connect gateway that are waiting for the client to send a request.

**Element identifier**  
gw\_cons\_wait\_client

**Element type**  
gauge

*Table 706. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcс_dbase             | Basic          |

**Usage** This value can change frequently. It should be sampled at regular intervals over an extended period in order to obtain a realistic view of gateway usage.

---

## gw\_cons\_wait\_host - Number of Connections Waiting for the Host to Reply

The current number of connections to host databases being handled by the DB2 Connect gateway that are waiting for a reply from the host.

**Element identifier**  
gw\_cons\_wait\_host

**Element type**  
gauge

*Table 707. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcс_dbase             | Basic          |

**Usage** This value can change frequently. It should be sampled at regular intervals over an extended period in order to obtain a realistic view of gateway usage.

---

## gw\_cur\_cons - Current Number of Connections for DB2 Connect

The current number of connections to host databases being handled by the DB2 Connect gateway.

*Table 708. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcs_dbase             | Basic          |

**Usage** This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

---

## gw\_db\_alias - Database Alias at the Gateway

The alias used at the DB2 Connect gateway to connect to the host database.

**Element identifier**

gw\_db\_alias

**Element type**

information

*Table 709. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Use this element for problem determination on DCS applications.

---

## gw\_exec\_time - Elapsed Time Spent on DB2 Connect Gateway Processing

The time in seconds and microseconds at the DB2 Connect gateway to process an application request (since the connection was established), or to process a single statement.

*Table 710. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch       |
|-----------------|-----------------------|----------------------|
| DCS Application | dcs_appl              | Statement, Timestamp |
| DCS Statement   | dcs_stmt              | Statement, Timestamp |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine what portion of the overall processing time is due to DB2 Connect gateway processing.

---

## gw\_total\_cons - Total Number of Attempted Connections for DB2 Connect

The total number of connections attempted from the DB2 Connect gateway since the last db2start command or the last reset.

Table 711. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcs_dbase             | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

---

## hadr\_connect\_status - HADR Connection Status monitor element

The current high availability disaster recovery (HADR) connection status of the database.

Table 712. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

Table 713. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the HADR connection status of a database.

The data type of this element is integer.

If the database is in HADR primary or standby role, the value for this element is one of the following constants:

#### SQLM\_HADR\_CONN\_CONNECTED

The database is connected to its partner node.

#### SQLM\_HADR\_CONN\_DISCONNECTED

The database is not connected to its partner node.

#### SQLM\_HADR\_CONN\_CONGESTED

The database is connected to its partner node, but the connection is congested. A connection is congested when the TCP/IP socket connection between the primary-standby pair is still alive, but one end cannot send to the other end. For example, the receiving end is not receiving from the socket connection, resulting in a full TCP/IP send space. The reasons for network connection being congested include the following:

- The network is being shared by too many resources or the network is not fast enough for the transaction volume of the primary HADR node.
- The server on which the standby HADR node resides is not powerful enough to retrieve information from the communication subsystem at the necessary rate.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_connect\_time - HADR Connection Time monitor element

This monitor element can return one of the following values: high availability disaster recovery (HADR) connection time, HADR congestion time, or HADR disconnection time.

*Table 714. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine when the current HADR connection status began.

If the database is in HADR primary or standby role, the meaning of this element depends on the value of the **hadr\_connect\_status** element:

- If the value of the **hadr\_connect\_status** element is `SQLM_HADR_CONN_CONNECTED`, then this element shows connection time.
- If the value of the **hadr\_connect\_status** element is `SQLM_HADR_CONN_CONGESTED`, then this element shows the time when congestion began.
- If the value of the **hadr\_connect\_status** element is `SQLM_HADR_CONN_DISCONNECTED`, then this element shows disconnection time.

If there has been no connection since the HADR engine dispatchable unit (EDU) was started, connection status is reported as Disconnected and HADR EDU startup time is used for the disconnection time. Since HADR connect and disconnect events are relatively infrequent, the time is collected and reported even if the `DFT_MON_TIMESTAMP` switch is off.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_heartbeat - HADR Heartbeat monitor element

Number of consecutively missed heartbeats on the high availability disaster recovery (HADR) connection.

This number is reset to zero when the database receives a heartbeat again. If the database is in the HADR primary or standby role, this element indicates the health of the HADR connection.

Table 715. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

For snapshot monitoring, this counter cannot be reset.

### Usage note:

Use this element to determine the health of the HADR connection.

A heartbeat is a message sent from the other HADR database at regular intervals. If the value for this element is zero, no heartbeats have been missed and the connection is healthy. The higher the value, the worse the condition of the connection.

In disconnected mode, heartbeat missed is always shown as 0, because it is not applicable.

The heartbeat interval is derived from configuration parameters such as **hadr\_timeout** and **hadr\_peer\_window**, with a maximal setting of 30 seconds.

The data type of this element is integer.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

Ignore this element if the HADR role of the database is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_local\_host - HADR Local Host monitor element

The local high availability disaster recovery (HADR) host name. The value is displayed as a host name string or an IP address string such as "1.2.3.4".

Table 716. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the effective HADR local host name. HADR database configuration parameters are static. Changes to a parameter are not

effective until the database is stopped and restarted. This monitor element reports the value that the HADR system is actually using rather than the value in the database configuration file.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Note:** Any name used must resolve to one IP address. A name that resolves to more than one address will cause an error when trying to start HADR.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_local\_service - HADR Local Service monitor element

The local HADR TCP service. This value is displayed as a service name string or a port number string.

Table 717. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the effective HADR local service name.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_log\_gap - HADR Log Gap

This element shows the running average of the gap between the primary Log sequence number (LSN) and the standby log LSN. The gap is measured in number of bytes.

Table 718. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

Table 719. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the gap between the primary and standby HADR database logs.

When a log file is truncated, the LSN in the next log file starts as if the last file were not truncated. This LSN hole does not contain any log data. Such holes can cause the log gap not to reflect the actual log difference between the primary and the standby HADR database logs.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_peer\_window - HADR peer window monitor element

The value of the HADR\_PEER\_WINDOW database configuration parameter.

Table 720. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the value of the HADR\_PEER\_WINDOW database configuration parameter.

---

## hadr\_peer\_window\_end - HADR peer window end monitor element

The point in time until which a high availability disaster recovery (HADR) primary database promises to stay in peer or disconnected peer state, as long as the primary database is active.

Table 721. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the point in time until which the primary promises to stay in peer or disconnected peer state.

The value reported by the primary database might be different from the value reported by the standby database. This occurs because the primary database

updates the value when it sends a heartbeat message, but the new value is shown on the standby database only after the message is received and processed on the standby database.

If a database moves out of peer or disconnected peer state, the value of this monitor element is not reset. The last known value is kept and returned. If a database never reached peer state, a value of zero will be returned.

The peer window end time is set by the primary database and then sent to the standby database. For this reason, the value of the peer window end is based on the clock of the primary database. When you compare the peer window end time with the primary database down time, you might need to add an offset to convert the timestamp to the primary database clock, if the two clocks are not well synchronized.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

---

## hadr\_primary\_log\_file - HADR Primary Log File monitor element

The name of the current log file on the primary HADR database.

Table 722. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log file on the primary HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_primary\_log\_lsn - HADR Primary Log LSN monitor element

The current log position of the primary HADR database. Log sequence number (LSN) is a byte offset in the database's log stream.

Table 723. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log position on the primary HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_primary\_log\_page - HADR Primary Log Page monitor element

The page number in the current log file indicating the current log position on the primary HADR database. The page number is relative to the log file. For example, page zero is the beginning of the file.

Table 724. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log page on the primary HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_remote\_host - HADR Remote Host monitor element

The remote high availability disaster recovery (HADR) host name. The value is displayed as a host name string or an IP address string such as "1.2.3.4".

Table 725. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the effective HADR remote host name.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Note:** Any name used must resolve to one IP address. A name that resolves to more than one address will cause an error when trying to start HADR.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_remote\_instance - HADR Remote Instance monitor element

The remote HADR instance name.

Table 726. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the effective HADR remote instance name.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_remote\_service - HADR Remote Service monitor element

The remote HADR TCP service. This value is displayed as a service name string or a port number string.

Table 727. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the effective HADR remote service name.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_role - HADR Role

The current high availability disaster recovery (HADR) role of the database.

Table 728. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

Table 729. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the HADR role of a database.

The data type of this element is integer.

The value for this element is one of the following constants:

#### SQLM\_HADR\_ROLE\_STANDARD

The database is not an HADR database.

#### SQLM\_HADR\_ROLE\_PRIMARY

The database is the primary HADR database.

#### SQLM\_HADR\_ROLE\_STANDBY

The database is the standby HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

---

## hadr\_standby\_log\_file - HADR Standby Log File monitor element

The name of the current log file on the standby HADR database.

Table 730. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log file on the standby HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_standby\_log\_lsn - HADR Standby Log LSN monitor element

The current log position of the standby HADR database. Log sequence number (LSN) is a byte offset in the database's log stream.

Table 731. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log position on the standby HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What's New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_standby\_log\_page - HADR Standby Log Page monitor element

The page number in the current log file indicating the current log position on the standby HADR database. The page number is relative to the log file. For example, page zero is the beginning of the file.

Table 732. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

### Usage

Use this element to determine the current log page on the standby HADR database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What's New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_state - HADR State monitor element

The current high availability disaster recovery (HADR) state of the database.

Table 733. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

Table 734. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the HADR state of a database.

The data type of this element is integer. If the database is in HADR primary or standby role, the value for this element is one of the following constants:

### SQLM\_HADR\_STATE\_DISCONNECTED

The database is not connected to its partner database.

### SQLM\_HADR\_STATE\_LOC\_CATCHUP

The database is doing local catchup.

### SQLM\_HADR\_STATE\_REM\_CATCH\_PEND

The database is waiting to connect to its partner to do remote catchup.

### SQLM\_HADR\_STATE\_REM\_CATCHUP

The database is doing remote catchup.

### SQLM\_HADR\_STATE\_PEER

The primary and standby databases are connected and are in peer state.

### SQLM\_HADR\_STATE\_DISCONN\_PEER

The primary and standby databases are in disconnected peer state.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see "Some monitoring interfaces for HADR have been deprecated" in *What's New for DB2 Version 10.1*.

---

## hadr\_syncmode - HADR Synchronization Mode monitor element

The current high availability disaster recovery (HADR) synchronization mode of the database.

Table 735. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

Table 736. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the HADR synchronization mode of a database.

The data type of this element is integer.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

If the database is in HADR primary or standby role, the value for this element is one of the following constants:

**SQLM\_HADR\_SYNCMODE\_SYNC**  
SYNC mode.

**SQLM\_HADR\_SYNCMODE\_NEARSYNC**  
NEARSYNC mode.

**SQLM\_HADR\_SYNCMODE\_ASYNC**  
ASYNC mode.

**SQLM\_HADR\_SYNCMODE\_SUPERASYNC**  
SUPERASYNC mode.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What's New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hadr\_timeout - HADR Timeout monitor element

The number of seconds without any communication from its partner after which an HADR database server will consider that the connection between them has failed.

*Table 737. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information | ACTIVITY METRICS BASE            |

*Table 738. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the effective HADR timeout value.

Changes to this element take effect on database activation or, if the database is already online, after HADR has been stopped and restarted on the primary.

**Important:** This monitor element has been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Some monitoring interfaces for HADR have been deprecated” in *What’s New for DB2 Version 10.1*.

This element should be ignored if the database's HADR role is standard. Use the **hadr\_role** monitor element to determine the HADR role of the database.

---

## hash\_join\_overflows - Hash Join Overflows

The number of times that hash join data exceeded the available sort heap space.

**Element identifier**

hash\_join\_overflows

**Element type**

counter

*Table 739. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 740. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** At the database level, if the value of hash\_join\_small\_overflows is greater than 10% of this hash\_join\_overflows, then you should consider increasing the sort heap size. Values at the application level can be used to evaluate hash join performance for individual applications.

---

## hash\_join\_small\_overflows - Hash Join Small Overflows

The number of times that hash join data exceeded the available sort heap space by less than 10%.

**Element identifier**

hash\_join\_small\_overflows

**Element type**

counter

*Table 741. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 742. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** If this value and `hash_join_overflows` are high, then you should consider increasing the sort heap threshold. If this value is greater than 10% of `hash_join_overflows`, then you should consider increasing the sort heap size.

---

## histogram\_type - Histogram type monitor element

The type of the histogram, in string format.

There are seven histogram types.

### **CoordActQueueTime**

A histogram of the time non-nested activities spend queued (for example, in a threshold queue), measured on the coordinator member.

### **CoordActExecTime**

A histogram of the time non-nested activities spend executing at the coordinator member. Execution time does not include time spent initializing or queued. For cursors, execution time includes only the time spent on open, fetch and close requests. When an activity is remapped between service subclasses, the execution time histogram is updated only for the service subclass in which the activity completes execution.

### **CoordActLifetime**

A histogram of the elapsed time from when a non-nested activity is identified by the database manager until the activity completes execution, as measured on the coordinator member. When you remap activities between service subclasses, the lifetime histogram is updated only for the service subclass in which the activity completes execution.

### **CoordActInterArrivalTime**

A histogram of the time interval between the arrival of non-nested coordinator activities. The inter-arrival time mean is calculated for service subclasses through which activities enter the system. When you remap activities between service subclasses, the inter-arrival time histogram of the service subclass you remap an activity to is unaffected.

### **CoordActEstCost**

A histogram of the estimated cost of non-nested DML activities. The estimated cost of an activity is counted only toward the service subclass in which the activity enters the system.

### **ReqExecTime**

A histogram of request execution times, which includes requests on the coordinator member, and any subrequests on both coordinator and non-coordinator members (like RPC requests or SMP subagent requests). Requests included may or may not be associated with an activity: Both PREPARE and OPEN requests are included in this histogram, for example, but while OPEN requests are always associated with a cursor activity, PREPARE requests are not part of any activity. The execution time histogram of a service subclass involved in remapping counts the portion of the execution time spent by the partial request in the service subclass.

### UowLifetime

A histogram of the elapsed time, measured in milliseconds, from the time that a unit of work is identified by the database manager until the time that the unit of work completes execution (committed or rolled back).

Table 743. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_histogrambin    | Always collected |

### Usage

Use this element to identify the type of histogram. Several histograms can belong to the same statistics record, but only one of each type.

---

## hld\_application\_handle - Identifier for the application holding the lock monitor element

System-wide unique ID for the application that is holding the lock. If the application holding this lock is unknown or cannot be found, then a value of NULL is returned.

Table 744. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

---

## hld\_member - Database member for application holding lock

Database member where the lock is being held by the holding application.

Table 745. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

### Usage

If the lock is being held on a remote member, the value of **hld\_member** is -2. To determine which member the lock is being held at, use the **MON\_GET\_LOCKS** table function and specify the **lock\_name** as a search argument.

---

## host\_ccsid - Host Coded Character Set ID

This is the coded character set identifier (CCSID) of the host database.

**Element identifier**  
host\_ccsid

**Element type**  
information

Table 746. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Use this element for problem determination on DCS applications.

## host\_db\_name - Host Database Name

The real name of the host database for which information is being collected or to which the application is connected. This is the name that was given to the database when it was created.

Table 747. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Database    | dcs_dbase             | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Use this element for problem determination on DCS applications.

## hostname - Host name monitor element

The host name of the machine on which the database member resides.

Table 748. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| DB2_CLUSTER_HOST_STATE administrative view and DB2_GET_CLUSTER_HOST_STATE table function - get information about hosts | ACTIVITY METRICS BASE            |
| MON_GET_CF_CMD table function - Get cluster caching facility command processing time                                   | ACTIVITY METRICS BASE            |
| MON_GET_CF_WAIT_TIME table function - Get cluster caching facility command wait time                                   | ACTIVITY METRICS BASE            |
| MON_GET_FCM - Get FCM metrics  | ACTIVITY METRICS BASE            |
| MON_GET_SERVERLIST table function - get member priority details  | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS - Get service subclass metrics  | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD - Get workload metrics  | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics  | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample workload metrics  | ACTIVITY METRICS BASE            |

---

## host\_name - Host name monitor element

Name of the host on which the cluster caching facility process resides.

*Table 749. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| DB_MEMBERS table function   | ACTIVITY METRICS BASE            |
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |
| ENV_GET_SYSTEM_RESOURCES table function - Return system information           | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics                                    | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information              | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function - Get memory set information                | ACTIVITY METRICS BASE            |

---

## host\_prdid - Host Product/Version ID

The product and version that is running on the server.

*Table 750. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Used to identify the product and code version of the DRDA host database product. It is in the form PPPVRRM, where:

- PPP identifies the host DRDA product
  - ARI for DB2 Server for VSE & VM
  - DSN for DB2 for z/OS
  - QSQ for DB2 for i
  - SQL for other DB2 products.
- VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)
- RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)
- M identifies a 1-character modification level (0-9 or A-Z)

---

## host\_response\_time - Host Response Time

At the DCS statement level, this is the elapsed time between the time that the statement was sent from the DB2 Connect gateway to the host for processing and the time when the result was received from the host. At DCS database and DCS application levels, it is the sum of the elapsed times for all the statements that were executed for a particular application or database. At the data transmission level, this is the sum of host response times for all the statements that used this many data transmissions.

Table 751. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dc_s_dbase            | Statement            |
| DCS Application   | dc_s_appl             | Statement, Timestamp |
| DCS Statement     | dc_s_stmt             | Statement, Timestamp |
| Data Transmission | stmt_transmissions    | Statement, Timestamp |

For snapshot monitoring at the statement level, this counter cannot be reset. This counter can be reset at other levels.

## Usage

Use this element with Outbound Number of Bytes Sent and Outbound Number of Bytes Received to calculate the outbound response time (transfer rate):

$$(\text{outbound bytes sent} + \text{outbound bytes received}) / \text{host response time}$$

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## id - cluster caching facility identification monitor element

cluster caching facility identifier, as defined in the db2nodes.cfg file.

Table 752. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| DB2_MEMBER and DB2_CF administrative views and DB2_GET_INSTANCE_INFO table function  | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CF_CMD table function - Get cluster caching facility command processing time | ACTIVITY METRICS BASE            |
| MON_GET_CF_WAIT_TIME table function - Get cluster caching facility command wait time | ACTIVITY METRICS BASE            |

---

## idle\_agents - Number of Idle Agents

The number of agents in the agent pool that are currently unassigned to an application and are, therefore, "idle".

Table 753. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

**Usage** You can use this element to help set the *num\_poolagents* configuration parameter. Having idle agents available to service requests for agents can improve performance.

---

## iid - Index identifier monitor element

Identifier for the index.

*Table 754. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO<br>table function - returns compressed index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function -<br>returns index information                       | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index<br>metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table<br>function - Get buffer pool page waiting<br>information | ACTIVITY METRICS BASE            |

---

## inbound\_bytes\_received - Inbound Number of Bytes Received

The number of bytes received by the DB2 Connect gateway from the client, excluding communication protocol usage (for example, TCP/IP).

*Table 755. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dc_s_appl             | Basic          |
| DCS Statement   | dc_s_stmt             | Statement      |

For snapshot monitoring at the application level, this counter can be reset. This counter cannot be reset at other levels.

**Usage** Use this element to measure the throughput from the client to the DB2 Connect gateway.

---

## inbound\_bytes\_sent - Inbound Number of Bytes Sent

The number of bytes sent by the DB2 Connect gateway to the client, excluding communication protocol usage (for example, TCP/IP).

*Table 756. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dc_s_appl             | Basic          |
| DCS Statement   | dc_s_stmt             | Statement      |

For snapshot monitoring at the application level, this counter can be reset. This counter cannot be reset at other levels.

**Usage** Use this element to measure the throughput from the DB2 Connect gateway to the client.

---

## inbound\_comm\_address - Inbound Communication Address

This is the communication address of the client. For example, it could be an IP address and port number for TCP/IP.

*Table 757. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Use this element for problem determination on DCS applications.

---

## include\_col\_updates - Include column updates monitor element

Number of include column updates.

*Table 758. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## incremental\_bind - Incremental bind monitor element

Indicates whether the package was incrementally bound at execution time. Possible values are YES or NO.

*Table 759. Event Monitoring Information*

| Event Type | Logical Data Grouping       | Monitor Switch |
|------------|-----------------------------|----------------|
| Locking    | lock_participant_activities |                |

---

## index\_jump\_scans - Index jump scans monitor element

Number of jump scans. A jump scan is an index scan where there is a gap in the index start and stop keys and sections of the index that will not yield results are skipped.

*Table 760. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

### Usage

You can use this monitor element to validate that jump scans are happening at an expected rate. If the value for this monitor element is not increasing as expected, check the following conditions:

- For queries being issued where you expect a jump scan, verify that the target table has an appropriate composite index and that the query predicates introduce an index gap. The DB2 optimizer will not create plans with jump scans if there are no index gaps.
- Jump scans do not scan the following types of indexes:

- range-clustered table indexes
- extended indexes (for example, spatial indexes)
- XML indexes
- text indexes (for Text Search)

**Note:** When creating an access plan, the DB2 optimizer performs costing analysis to determine if jump scans should be executed. There might be cases where the optimizer finds that not using a jump scan is more efficient.

---

## index\_name - Index name monitor element

The name of an index.

*Table 761. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO<br>table function - returns compressed index information       | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function -<br>returns index information                           | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |

---

## index\_schema - Index schema monitor element

The name of an index schema.

*Table 762. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO<br>table function - returns compressed index<br>information    | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function -<br>returns index information                           | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |

---

## index\_object\_pages - Index Object Pages

The number of disk pages consumed by all indexes defined on a table.

*Table 763. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 764. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

**Usage** This element provides a mechanism for viewing the actual amount of space consumed by indexes defined on a particular table. This element can be used in conjunction with a table event monitor to track the rate of index growth over time. This element is not returned for partitioned tables.

---

## index\_object\_l\_pages - Index data logical pages monitor element

The number of logical pages used on disk by all indexes associated with this table. For partitioned tables, the value returned is NULL.

*Table 765. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

### Usage

- This value might be less than the amount of space physically allocated for the object. This can happen when you use the REUSE STORAGE option of the TRUNCATE statement. This option causes storage allocated for the table to continue to be allocated, although the storage will be considered empty. In addition, the value for this monitor element might be less than the amount of space logically allocated for the object, because the total space logically allocated includes a small amount of additional meta data.

To retrieve an accurate measure of the logical or physical size of an object, use the ADMIN\_GET\_TAB\_INFO\_V97 function. This function provides more accurate information about the size of objects than you can obtain by multiplying the number of pages reported for this monitor element by the page size.

---

## index\_only\_scans - Index-only scans monitor element

Number of index-only scans. Index-only scans occur when the results of scan was satisfied by access to the index only.

*Table 766. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## index\_scans - Index scans monitor element

Number of index scans.

*Table 767. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## index\_tbsp\_id - Index table space ID monitor element

An identifier of the table space that holds indexes created on this table.

Table 768. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE                        |

### Usage

The value of this element matches a value from column TBSPACEID of view SYSCAT.TABLESPACES.

---

## input\_db\_alias - Input Database Alias

The alias of the database provided when calling the snapshot function.

Table 769. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl_id_info          | Basic          |
| Table Space    | tablespace_list       | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Table          | table_list            | Table          |
| Lock           | db_lock_list          | Basic          |

**Usage** This element can be used to identify the specific database to which the monitor data applies. It contains blanks unless you requested monitor information related to a specific database.

The value of this field may be different than the value of the *client\_db\_alias* monitor element since a database can have many different aliases. Different applications and users can use different aliases to connect to the same database.

---

## insert\_sql\_stmts - Inserts

This element contains a count of the total number of times the federated server has issued an INSERT statement to this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

Table 770. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to determine the level of database activity directed against this data source by the federated server or an application.

You can also use this element to determine the percentage of write activity against this data source by the federated server or an application, with the following formula:

```
write_activity =  
    (INSERT statements + UPDATE statements + DELETE statements ) /  
    (SELECT statements + INSERT statements + UPDATE statements +  
    DELETE statements)
```

---

## insert\_time - Insert Response Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to INSERTs from all applications or a single application running on this federated server instance since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

The response time is measured as the difference in time between the time the federated server submits an INSERT statement to the data source, and the time the data source responds to the federated server, indicating that the INSERT has been processed.

*Table 771. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to determine the actual amount of time that transpires waiting for INSERTs to this data source to be processed. This information can be useful for capacity planning and tuning.

---

## insert\_timestamp - Insert timestamp monitor element

The time when the statement or section was inserted into the cache. For dynamic sql snapshots this represents the time when the statement entered the cache. For MON\_GET\_PKG\_CACHE\_STMT, MON\_GET\_PKG\_CACHE\_STMT\_DETAILS and the package cache event monitor, the value is more granular and represents the time when an individual section for this statement was inserted into the cache.

*Table 772. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache | ACTIVITY METRICS BASE            |

Table 772. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element  | Collection Level |
|---|------------------|------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS                                  | ACTIVITY METRICS | BASE             |
| table function - get detailed metrics for package cache entries |                  |                  |

Table 773. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

Table 774. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

## Usage

This element specifies the time when the statement was inserted into the cache. It can be used to estimate the lifetime of a statement in the cache.

## int\_auto\_rebinds - Internal Automatic Rebinds

The number of automatic rebinds (or recompiles) that have been attempted.

Table 775. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 776. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Automatic rebinds are the internal binds the system performs when a package has been invalidated. The rebind is performed the first time that the database manager needs to execute an SQL statement from the package. For example, packages are invalidated when you:

- Drop an object, such as a table, view, or index, on which the plan is dependent
- Add or drop a foreign key
- Revoke object privileges on which the plan is dependent.

You can use this element to determine the level of database activity at the application or database level. Since int\_auto\_rebinds can have a significant impact on performance, they should be minimized where possible.

You can also use this element to determine the percentage of rebind activity using the following formula:

int\_auto\_rebinds / total number of statements

This information can be useful for analyzing application activity and throughput.

---

## int\_commits - Internal commits monitor element

The total number of commits initiated internally by the database manager.

*Table 777. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 778. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 779. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch       |
|------------|---|----------------------|
| Database   | event_db  | Always collected     |
| Connection | event_conn  | Always collected     |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |

Table 779. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## Usage

An internal commit may occur during any of the following events:

- A reorganization
- An import
- A bind or pre-compile
- An application ends without executing an explicit SQL COMMIT statement (on UNIX).

This value, which does not include explicit SQL COMMIT statements, represents the number of these internal commits since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

You can use this element to calculate the total number of units of work by calculating the sum using the following expression:

```

    commit_sql_stmts
+   int_commits
+   rollback_sql_stmts
+   int_rollbacks

```

**Note:** The units of work calculated will only include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at the application or the database level.

---

## int\_deadlock\_rollbacks - Internal Rollbacks Due To Deadlock

The total number of forced rollbacks initiated by the database manager due to a deadlock. A rollback is performed on the current unit of work in an application selected by the database manager to resolve the deadlock.

### Element identifier

int\_deadlock\_rollbacks

### Element type

counter

Table 780. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 780. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 781. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** This element shows the number of deadlocks that have been broken and can be used as an indicator of concurrency problems. It is important, since `int_deadlock_rollbacks` lower the throughput of the database.

This value is included in the value given by `int_rollbacks`.

---

## int\_node\_splits - Intermediate node splits monitor element

Number of times an intermediate index node was split during an insert operation.

Table 782. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## int\_rollbacks - Internal rollbacks monitor element

The total number of rollbacks initiated internally by the database manager.

Table 783. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 783. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 784. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 785. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Database     | event_db   | Always collected     |
| Connection   | event_conn   | Always collected     |
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## Usage

An internal rollback occurs when any of the following cannot complete successfully:

- A reorganization
- An import
- A bind or pre-compile
- An application ends as a result of a deadlock situation or lock timeout situation
- An application ends without executing an explicit commit or rollback statement (on Windows).

This value represents the number of these internal rollbacks since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

While this value does not include explicit SQL ROLLBACK statements, the count from the **int\_deadlock\_rollbacks** monitor element is included.

You can use this element to calculate the total number of units of work by calculating the sum of the following:

```

    commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks

```

**Note:** The units of work calculated will include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at the application or the database level.

---

## int\_rows\_deleted - Internal Rows Deleted

This is the number of rows deleted from the database as a result of internal activity.

*Table 786. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

*Table 787. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |
| Statements | event_stmt            | Always collected |

**Usage** This element can help you gain insight into internal activity within the database manager of which you might not be aware. If this activity is high, you may want to evaluate your table design to determine if the referential constraints or triggers that you have defined on your database are necessary.

Internal delete activity can be a result of:

- A cascading delete enforcing an ON CASCADE DELETE referential constraint
- A trigger being fired.

---

## int\_rows\_inserted - Internal Rows Inserted

The number of rows inserted into the database as a result of internal activity caused by triggers.

*Table 788. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 788. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

Table 789. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |
| Statements | event_stmt            | Always collected |

**Usage** This element can help you gain insight into the internal activity within the database manager. If this activity is high, you may want to evaluate your design to determine if you can alter it to reduce this activity.

---

## int\_rows\_updated - Internal Rows Updated

This is the number of rows updated from the database as a result of internal activity.

Table 790. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

Table 791. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |
| Statements | event_stmt            | Always collected |

**Usage** This element can help you gain insight into internal activity within the database manager of which you might not be aware. If this activity is high, you may want to evaluate your table design to determine if the referential constraints that you have defined on your database are necessary.

Internal update activity can be a result of:

- A *set null* row update enforcing a referential constraint defined with the ON DELETE SET NULL rule
- A trigger being fired.

---

## intra\_parallel\_state - Current state of intrapartition parallelism monitor element

The current state of intrapartition parallelism reported at statement, activity, transaction, or workload level. Possible values are "YES" to indicate that intrapartition parallelism is enabled so a statement can run with subagents; and "NO" to indicate that intrapartition parallelism is disabled.

Table 792. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                                | ACTIVITY METRICS BASE            |

Table 793. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Activities   | -                     | Always collected |
| Locking      |                       |                  |
| Unit of work |                       |                  |

---

## invocation\_id - Invocation ID monitor element

An identifier that distinguishes one invocation of a routine from others at the same nesting level within a unit of work. It is unique within a unit of work for a specific nesting level.

The **invocation\_id** monitor element is an alias of the **stmt\_invocation\_id** monitor element.

Table 794. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 795. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activitystmt    | -              |
| Locking    | -                     | -              |

Table 795. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping         | Monitor Switch |
|--|-------------------------------|----------------|
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history            | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history            | -              |
| Unit of work                                       | Reported in the package list. | -              |

- 1 This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element to uniquely identify the invocation in which a particular SQL statement has been executed. You can also use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

---

## ipc\_rcv\_volume - Interprocess communication received volume monitor element

The amount of data received by data server from clients over IPC. This value is reported in bytes.

Table 796. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 796. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 797. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## ipc\_recv\_wait\_time - Interprocess communication received wait time monitor element

The time spent by an agent receiving an incoming client request using the IPC communications protocol. The value is reported in milliseconds.

Table 798. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 798. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 799. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## ipc\_recvs\_total - Interprocess communication receives total monitor element

The number of times data was received by the database server from the client application using IPC.

Table 800. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 801. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## ipc\_send\_volume - Interprocess communication send volume monitor element

The amount of data sent by data server to clients over the IPC protocol. This value is reported in bytes.

Table 802. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 803. Event Monitoring Information

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |

Table 803. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                    | Monitor Switch |
|--------------|--|----------------|
| Unit of work | Reported in the system_metrics document. | -              |

## ipc\_send\_wait\_time - Interprocess communication send wait time monitor element

The time spent blocking on an IPC send to the client. The value is given in milliseconds.

Table 804. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 805. Event Monitoring Information

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |

Table 805. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## ipc\_sends\_total - Interprocess communication send total monitor element

The number of times data was sent by the database server to client applications using IPC.

Table 806. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 807. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

---

## is\_system\_appl - Is System Application monitor element

Indicates whether the application is a system application.

Table 808. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |

### Usage

The **is\_system\_appl** monitor element indicates whether an application is an internal system application. The possible values are

- 0 user application
- 1 system application

---

## key\_updates - Key updates monitor element

Number of key updates.

Table 809. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## last\_active\_log - Last Active Log File Number

The file number of the last active log file.

### Element identifier

last\_active\_log

### Element type

information

Table 810. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

Table 811. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 812. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *first\_active\_log* and *current\_active\_log* elements to determine the range of active log files. Knowing the range of active log files helps you determine the disk space required for log files.

You can also use this element to determine which log files have data to help you identify log files needed for split mirror support.

---

## last\_backup - Last Backup Timestamp

The date and time that the latest database backup was completed.

**Element identifier**

last\_backup

**Element type**

timestamp

*Table 813. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Timestamp      |

**Usage** You may use this element to help you identify a database that has not been backed up recently, or to identify which database backup file is the most recent. If the database has never been backed up, this timestamp is initialized to zero.

---

## last\_executable\_id - Last executable identifier monitor element

The executable id for the statement most recently completed by the application.

*Table 814. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

---

## last\_extent - Last extent moved monitor element

The numeric identifier of the last extent moved by the table space rebalancer process.

*Table 815. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_EXTENT_MOVEMENT_STATUS table function - Get extent movement progress status metrics | ACTIVITY METRICS BASE            |

---

## last\_metrics\_update - Metrics last update timestamp monitor element

Timestamp reflecting the last time metrics were updated for this cache entry.

Table 816. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 817. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

---

## last\_overflow\_time - Time of Last Event Overflow

The date and time of the last overflow recorded this overflow record.

Table 818. Event Monitoring Information

| Event Type      | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Overflow Record | event_overflow        | -              |

**Usage** Use this element with *first\_overflow\_time* to calculate the elapsed time for which the overflow record was generated.

---

## last\_reference\_time - Last reference time monitor element

The last time the activity was accessed by a request.

Table 819. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

### Usage

---

## last\_request\_type - Last request type monitor element

The type of the last request completed by the application.

Table 820. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 820. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

This monitor element is only reported on the coordinator member for the application.

The following values are possible.

- CLOSE
- COMMIT
- COMPILE
- DESCRIBE
- EXCSQLSET
- EXECIMMD
- EXECUTE
- FETCH
- INTERNAL *number*, where *number* is the value of the internal constant
- OPEN
- PREPARE
- REBIND
- REDISTRIBUTE
- REORG
- ROLLBACK
- RUNSTATS

---

## last\_reset - Last Reset Timestamp

Indicates the date and time that the monitor counters were reset for the application issuing the GET SNAPSHOT.

### Element identifier

last\_reset

### Element type

timestamp

Table 821. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics                  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses | ACTIVITY METRICS BASE            |

Table 821. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUPERCLASS_STATS<br>table function - Return statistics of service superclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS<br>table function - Return work action set statistics            | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                          | ACTIVITY METRICS BASE            |

Table 822. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch         |
|------------------|-----------------------|------------------------|
| Database Manager | db2                   | Timestamp              |
| Database         | dbase                 | Timestamp              |
| Application      | appl                  | Timestamp              |
| Table Space      | tablespace_list       | Buffer Pool, Timestamp |
| Table            | table_list            | Timestamp              |
| DCS Database     | dcs_dbase             | Timestamp              |
| DCS Application  | dcs_appl              | Timestamp              |

**Usage** You can use this element to help you determine the scope of information returned by the database system monitor.

If the counters have never been reset, this element will be zero.

The database manager counters will only be reset if you reset all active databases.

---

## last\_updated - Last update time stamp monitor element

The time stamp indicating the last time this entry was updated.

Table 823. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table usage list  | ACTIVITY METRICS BASE            |

---

## last\_wlm\_reset - Time of last reset monitor element

This element, in the form of a local timestamp, shows the time at which the last statistics event record of this type was created.

Table 824. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

Table 824. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_qstats          | -              |

## Usage

Use the **wlm\_last\_reset** and **statistics\_timestamp** monitor elements to determine a period of time over which the statistics in an event monitor statistics record were collected. The collection interval begins at the **wlm\_last\_reset** time and ends at **statistics\_timestamp**.

## lob\_object\_pages - LOB Object Pages

The number of disk pages consumed by LOB data.

Table 825. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 826. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

**Usage** This element provides a mechanism for viewing the actual amount of space consumed by LOB data in a particular table. This element can be used in conjunction with a table event monitor to track the rate of LOB data growth over time.

## lob\_object\_l\_pages - LOB data logical pages monitor element

The number of logical pages used on disk by LOBs associated with this table.

Table 827. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

## Usage

- This value might be less than the amount of space physically allocated for the object. This can happen when you use the REUSE STORAGE option of the TRUNCATE statement. This option causes storage allocated for the table to continue to be allocated, although the storage will be considered empty. In addition, the value for this monitor element might be less than the amount of space logically allocated for the object, because the total space logically allocated includes a small amount of additional meta data.

To retrieve an accurate measure of the logical or physical size of an object, use the ADMIN\_GET\_TAB\_INFO\_V97 function. This function provides more

accurate information about the size of objects than you can obtain by multiplying the number of pages reported for this monitor element by the page size.

---

## local\_cons - Local Connections

The number of local applications that are currently connected to a database within the database manager instance being monitored.

*Table 828. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

This number can help you determine the level of concurrent processing occurring in the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage.

This number only includes applications that were initiated from the same instance as the database manager. The applications are connected, but may or may not be executing a unit of work in the database.

When used in conjunction with the `rem_cons_in` monitor element, this element can help you adjust the setting of the `max_connections` configuration parameter.

---

## local\_cons\_in\_exec - Local Connections Executing in the Database Manager

The number of local applications that are currently connected to a database within the database manager instance being monitored and are currently processing a unit of work.

*Table 829. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

This number can help you determine the level of concurrent processing occurring in the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number only includes applications that were initiated from the same instance as the database manager.

When used in conjunction with the `rem_cons_in_exec` monitor element, this element can help you adjust the setting of the `max_coordagents` configuration parameter.

The following recommendations apply to non-concentrator configurations only. When concentrator is enabled, DB2 is multiplexing a larger number of client

connections onto a smaller pool of coordinator agents. In this case, it is usually acceptable to have the sum of `rem_cons_in_exec` and `local_cons_in_exec` approach the `max_coordagents` value.

- If `max_coordagents` is set to AUTOMATIC, do not make any adjustments.
- If `max_coordagents` is not set to AUTOMATIC and if the sum of `rem_cons_in_exec` and `local_cons_in_exec` is close to `max_coordagents`, increase the value of `max_coordagents`.

---

## local\_start\_time - Local start time monitor element

The time that this activity began doing work on the member. It is in local time. This field can be an empty string when an activity has entered the system but is in a queue and has not started executing.

*Table 830. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES table function - Return a list of activities                       | ACTIVITY METRICS BASE            |

### Usage

---

## local\_transaction\_id - Local transaction identifier monitor element

The local transaction ID in use at the time the event occurred.

*Table 831. Event Monitoring Information*

| Event Type     | Logical Data Grouping        | Monitor Switch   |
|----------------|------------------------------|------------------|
| Change history | ddlstmtexec<br>txncompletion | Always collected |
| Unit of Work   | uow                          |                  |

### Usage

For the change history event monitor, this is the local transaction ID in use at the time the event occurred. This is the SQLU\_TID structure that is part of the transaction logs.

---

## location - Location

Identifies the location associated with the event.

*Table 832. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILLOCATION          | Always collected |

## Usage

For the change history event monitor, locations depend on the UTILITY\_TYPE, for example, load input files or backup target path name.

---

## location\_type - Location type

A description of what the location is used for.

Table 833. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILLOCATION          | Always collected |

## Usage

If the utility\_type element is LOAD, one of:

- C Copy target
- D Input data
- L LOB path
- X XML path

If the utility\_type element is BACKUP, one of:

- B Backup target location

If the utility\_type element is RESTORE, one of:

- S Restore source location

If the utility\_type element is ROLLFORWARD, one of:

- O Alternate overflow log path captured as part of the ROLLFORWARD DATABASE command. Note that if the default overflow log path is used, no location record will be captured.

otherwise a blank character.

---

## lock\_attributes - Lock attributes monitor element

The lock attributes of the application that is currently holding the lock.

Table 834. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE            |

Table 835. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 835. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock_wait             | Basic          |

Table 836. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

**1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The following table lists all possible lock attribute settings. Each lock attribute setting is based upon a bit flag value defined in sqlmon.h.

| Lock Attribute Value in Table Functions | API Constant                 | Description                            |
|---|------------------------------|--|
| 0000000000000001                        | SQLM_LOCKATTR_WAIT_FOR_AVAIL | Wait for availability.                 |
| 0000000000000002                        | SQLM_LOCKATTR_ESCALATED      | Acquired by escalation.                |
| 0000000000000004                        | SQLM_LOCKATTR_RR_IN_BLOCK    | RR lock in block.                      |
| 0000000000000008                        | SQLM_LOCKATTR_INSERT         | Insert lock.                           |
| 0000000000000010                        | SQLM_LOCKATTR_RR             | Lock by RR scan.                       |
| 0000000000000020                        | SQLM_LOCKATTR_UPDATE_DELETE  | Update/delete row lock.                |
| 0000000000000040                        | SQLM_LOCKATTR_ALLOW_NEW      | Allow new lock requests.               |
| 0000000000000080                        | SQLM_LOCKATTR_NEW_REQUEST    | A new lock requester.                  |
| 0000000000000200                        | SQLM_LOCKATTR_INDOUBT        | Lock held by Indoubt Transaction.      |
| 0000000000000400                        | SQLM_LOCKATTR_LOW_PRIORITY   | Lock held by low priority application. |

Bits returned that are not listed in the previously shown table are reserved for internal use.

## lock\_count - Lock count monitor element

The number of locks on the lock being held.

Table 837. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- Get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

Table 838. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 839. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This value ranges from 1 to 255. It is incremented as new locks are acquired, and decremented as locks are released.

When the **lock\_count** monitor element has a value of 255, this indicates that a transaction duration lock is being held. At this point, the **lock\_count** monitor element is no longer incremented or decremented when locks are acquired or released. The **lock\_count** monitor element is set to a value of 255 in one of two possible ways:

1. The **lock\_count** monitor element value is incremented 255 times due to new locks being acquired.
2. A transaction duration lock is explicitly acquired. For example, with a LOCK TABLE statement, or an INSERT.

---

## lock\_current\_mode - Original lock mode before conversion monitor element

During a lock conversion operation, the lock mode held by the application waiting to acquire the lock, before the conversion is completed.

Table 840. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE            |

Table 841. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Basic          |

Table 842. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

**1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The following scenario describes an example of lock conversion. During an update or delete operation it is possible to wait for an X lock on the target row. If the transaction is holding an S or V lock on the row, this would require a conversion. At this point, the **lock\_current\_mode** element is assigned a value of S or V, while the lock waits to be converted to an X lock.

The possible lock modes are listed in the following table.

| Mode | Type of Lock                        | API Constant |
|------|-------------------------------------|--------------|
|      | No Lock                             | SQLM_LNON    |
| IS   | Intention Share Lock                | SQLM_LOIS    |
| IX   | Intention Exclusive Lock            | SQLM_LOIX    |
| S    | Share Lock                          | SQLM_LOOS    |
| SIX  | Share with Intention Exclusive Lock | SQLM_LSIX    |
| X    | Exclusive Lock                      | SQLM_LOOX    |
| IN   | Intent None                         | SQLM_LOIN    |
| Z    | Super Exclusive Lock                | SQLM_LOOZ    |
| U    | Update Lock                         | SQLM_LOOU    |
| NS   | Scan Share Lock                     | SQLM_LONS    |
| NW   | Next Key Weak Exclusive Lock        | SQLM_LONW    |

## lock\_escalation - Lock escalation monitor element

Indicates whether the application waiting to acquire this lock was a result of a lock escalation request. The possible values are Y (Yes) and N (No).

Table 843. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |

Table 844. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 845. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Deadlocks <sup>1</sup>              | lock                  | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Use this element to better understand the cause of deadlocks. If you experience a deadlock that involves applications doing lock escalation, you may want to increase the amount of lock memory or change the percentage of locks that any one application can request.

---

## lock\_escals - Number of lock escalations monitor element

The number of times that locks have been escalated from several row locks to a table lock.

Table 846. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 846. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLE table function - get table metrics  | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list   | DATA OBJECT METRICS EXTENDED     |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 847. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 848. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Transactions  | event_xact  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the **maxlocks** and **locklist** configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs.

This data item includes a count of all lock escalations, including exclusive lock escalations and escalations in the DB2 pureScale environment. To determine just the lock escalations in the DB2 pureScale environment, use the **lock\_escals\_global** monitor element.

There are several possible causes for excessive lock escalations:

- The lock list size (**locklist**) may be too small for the number of concurrent applications
- The percent of the lock list usable by each application (**maxlocks**) may be too small
- One or more applications may be using an excessive number of locks.
- In the DB2 pureScale environment, the global lock list size (**cf\_lock\_sz**) may be too small.

To resolve these problems, you may be able to:

- Increase the **locklist** configuration parameter value.
- Increase the **maxlocks** configuration parameter value.
- Identify the applications with large numbers of locks, or those that are holding too much of the lock list, using one of the following formulae, and comparing the value to **maxlocks**.
  - On 64-bit systems,  $((locks\ held * 64) / (locklist * 4096)) * 100$
  - On 32-bit systems,  $((locks\ held * 48) / (locklist * 4096)) * 100$

These applications can also cause lock escalations in other applications by using too large a portion of the lock list. These applications may need to resort to using table locks instead of row locks, although table locks may cause an increase in **lock\_waits** and **lock\_wait\_time** monitor element values.

---

## lock\_escals\_global - Number of global lock escalations monitor element

Number of lock escalations on a global lock due to global lock memory usage reaching the limit specified in the **cf\_lock\_sz** database configuration parameter.

*Table 849. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 849. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS    | BASE             |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS     | BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS     | BASE             |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS | EXTENDED         |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS | EXTENDED         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS     | BASE             |

Table 850. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element together with the **lock\_escalations\_maxlocks** and **lock\_escalations\_locklist** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

---

### lock\_escalations\_locklist - Number of locklist lock escalations monitor element

Number of lock escalations due to local lock memory usage reaching the limit specified in the **locklist** database configuration parameter.

*Table 851. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 852. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 852. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

Use this monitor element together with the **lock\_escals\_maxlocks** and **lock\_escals\_global** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

## lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element

Number of lock escalations due to local lock memory usage reaching the limit specified in the **maxlocks** database configuration parameter.

Table 853. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 853. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element | Collection Level |
|--|-----------------|------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS | BASE             |

Table 854. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element together with the **lock\_escals\_locklist** and **lock\_escals\_global** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

## lock\_hold\_count - Lock hold count monitor element

The number of holds placed on the lock. Holds are placed on locks by cursors registered with the WITH HOLD clause and some DB2 utilities. Locks with holds are not released when transactions are committed.

Table 855. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 856. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

---

## lock\_list\_in\_use - Total lock list memory in use monitor element

The total amount of lock list memory (in bytes) that is in use.

Table 857. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

### Usage

This element may be used in conjunction with the **locklist** configuration parameter to calculate the lock list utilization. If the lock list utilization is high, you may want to consider increasing the size of that parameter.

**Note:** When calculating utilization, it is important to note that the **locklist** configuration parameter is allocated in pages of 4 KB each, while this monitor element provides results in bytes.

---

## lock\_mode - Lock mode monitor element

The type of lock being held. If the mode is unknown, the value of this monitor element is NULL.

Table 858. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE                        |

Table 859. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 860. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch |
|-------------------------------------|-----------------------|----------------|
| Locking                             | -                     | -              |
| Deadlocks <sup>1</sup>              | lock                  | -              |
| Deadlocks <sup>1</sup>              | event_dlconn          | -              |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | -              |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This mode can help you determine the source of contention for resources.

This element indicates one of the following, depending on the type of monitor information being examined:

- The type of lock another application holds on the object that this application is waiting to lock (for application-monitoring and deadlock-monitoring levels).
- The type of lock held on the object by this application (for object-lock levels).

The possible values for this field are:

| Mode | Type of Lock                        | API Constant |
|------|-------------------------------------|--------------|
|      | No Lock                             | SQLM_LNON    |
| IS   | Intention Share Lock                | SQLM_LOIS    |
| IX   | Intention Exclusive Lock            | SQLM_LOIX    |
| S    | Share Lock                          | SQLM_LOOS    |
| SIX  | Share with Intention Exclusive Lock | SQLM_LSIX    |
| X    | Exclusive Lock                      | SQLM_LOOX    |
| IN   | Intent None                         | SQLM_LOIN    |
| Z    | Super Exclusive Lock                | SQLM_LOOZ    |
| U    | Update Lock                         | SQLM_LOOU    |
| NS   | Scan Share Lock                     | SQLM_LONS    |
| NW   | Next Key Weak Exclusive Lock        | SQLM_LONW    |

---

## lock\_mode\_requested - Lock mode requested monitor element

The mode in which the lock was requested by the application waiting to acquire the lock.

*Table 861. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |

*Table 862. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock_wait             | Lock           |

*Table 863. Event Monitoring Information*

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The mode in which the lock was requested by the application. This value can help you determine the source of contention for resources.

The possible lock modes are listed in the following table.

| Mode | Type of Lock                        | API Constant |
|------|-------------------------------------|--------------|
|      | No Lock                             | SQLM_LNON    |
| IS   | Intention Share Lock                | SQLM_LOIS    |
| IX   | Intention Exclusive Lock            | SQLM_LOIX    |
| S    | Share Lock                          | SQLM_LOOS    |
| SIX  | Share with Intention Exclusive Lock | SQLM_LSIK    |
| X    | Exclusive Lock                      | SQLM_LOOX    |
| IN   | Intent None                         | SQLM_LOIN    |
| Z    | Super Exclusive Lock                | SQLM_LOOZ    |
| U    | Update Lock                         | SQLM_LOOU    |
| NS   | Scan Share Lock                     | SQLM_LONS    |
| NW   | Next Key Weak Exclusive Lock        | SQLM_LONW    |

---

## lock\_name - Lock name monitor element

Internal binary lock name. This element serves as a unique identifier for locks.

*Table 864. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE                        |

*Table 865. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | lock_wait      |

Table 866. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Locking                | -                     | -              |
| Deadlocks <sup>1</sup> | lock                  | -              |
| Deadlocks <sup>1</sup> | event_dlconn          | -              |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The internal name can be formatted using the routine MON\_FORMAT\_LOCK\_NAME to obtain more details about the lock. For example, if this is a table lock, then you can obtain the table and tablespace that the lock references.

---

## lock\_node - Lock Node

The node involved in a lock.

Table 867. Snapshot Monitoring Information

| Snapshot Level         | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Application            | subsection            | Statement      |
| Deadlocks              | event_dlconn          | Statement      |
| Deadlocks with Details | event_detailed_dlconn | Statement      |

**Usage** This can be used for troubleshooting.

---

## lock\_object\_name - Lock Object Name

This element is provided for informational purposes only. It is the name of the object for which the application holds a lock (for object-lock-level information), or the name of the object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

Table 868. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Basic          |

Table 869. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Deadlocks  | lock                  | Always collected |

Table 869. Event Monitoring Information (continued)

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** For table-level locks, it is the file ID (FID) for SMS and DMS table spaces. For row-level locks, the object name is the row ID (RID). For table space locks, the object name is blank. For buffer pool locks, the object name is the name of the buffer pool.

To determine the table holding the lock, use *table\_name* and *table\_schema* instead of the file ID, since the file ID may not be unique.

To determine the table space holding the lock, use *tablespace\_name*.

## lock\_object\_type - Lock object type waited on monitor element

The type of object against which the application holds a lock (for object-lock-level information), or the type of object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).

Table 870. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_FORMAT_LOCK_NAME table function - format the internal lock name and return details                 | ACTIVITY METRICS BASE                        |
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE                        |

Table 871. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Lock           |

Table 872. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch |
|-------------------------------------|-----------------------|----------------|
| Locking                             | -                     | -              |
| Deadlocks <sup>1</sup>              | lock                  | -              |
| Deadlocks <sup>1</sup>              | event_dlconn          | -              |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | -              |

**1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT

MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This element can help you determine the source of contention for resources.

For snapshot monitoring and the deadlock<sup>1</sup> event monitor, the object type identifiers are defined in sqlmon.h. The objects may be one of the following types:

- Table space (SQLM\_TABLESPACE\_LOCK in sqlmon.h)
- Table
- Buffer pool
- Block
- Record (or row)
- Data partition (SQLM\_TABLE\_PART\_LOCK in sqlmon.h)
- Internal (another type of lock held internally by the database manager)
- Automatic resize
- Automatic storage.

For the locking event monitor and the monitoring table functions in Table 1, the possible values for the **lock\_object\_type** monitor element are defined in Table 4.

*Table 873. Possible values for lock\_object\_type monitor element*

| Possible values | Description                 |
|-----------------|-----------------------------|
| TABLE           | Table lock                  |
| ROW             | Row lock                    |
| TABLESPACE      | Table space lock            |
| EOT             | End of table lock           |
| KEYVALUE        | Key value lock              |
| SYSBOOT         | Sysboot lock                |
| PLAN            | Plan lock                   |
| VARIATION       | Variation lock              |
| SEQUENCE        | Sequence lock               |
| BUFFERPOOL      | Buffer pool lock            |
| LOB             | LOB/Long region lock        |
| CATALOG         | Catalog cache lock          |
| ONLINE_BACKUP   | Online backup lock          |
| OBJECT_TABLE    | Object table lock           |
| ALTER_TABLE     | Table alter lock            |
| DMS_SEQUENCE    | DMS sequence lock           |
| REORG           | Inplace reorganization lock |
| MDC_BLOCK       | MDC block lock              |
| TABLE_PARTITION | Table partition lock        |
| AUTORESIZE      | Autoresize lock             |
| AUTOSTORAGE     | Autostorage lock            |
| XMLPATH         | XML path lock               |

Table 873. Possible values for lock\_object\_type monitor element (continued)

| Possible values  | Description                      |
|------------------|----------------------------------|
| EXTENT_MOVEMENT  | Extent movement lock             |
| WORKLOAD         | Workload authorization lock      |
| FED_SERVER       | Federation server lock           |
| FED_USER         | Federation user mapping lock     |
| CHUNK            | Chunk lock                       |
| LOAD_PRE_PART    | Load table pre-partitioning lock |
| LOAD_PART        | Load table partitioning lock     |
| LOAD_TS          | Loading table space lock         |
| LONG_FIELD_ESC   | Long field escalation lock       |
| LONG_FIELD_SPACE | Long field buddy space lock      |

## lock\_release\_flags - Lock release flags monitor element

Lock release flags.

Table 874. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_LOCKS table function - List all locks in the currently connected database | ACTIVITY METRICS BASE            |

Table 875. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Basic          |

Table 876. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

The following table lists all possible release flag settings. Each release flag is based upon a bit flag value defined in sqlmon.h.

| API Constant                   | Description                  |
|--------------------------------|------------------------------|
| SQLM_LOCKRELFIELDS_SQLCOMPILER | Locks by SQL compiler.       |
| SQLM_LOCKRELFIELDS_UNTRACKED   | Non-unique, untracked locks. |

**Note:** All non-assigned bits are used for application cursors.

## lock\_status - Lock status monitor element

Indicates the internal status of the lock.

*Table 877. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting | ACTIVITY METRICS BASE                        |
| MON_GET_LOCKS table function - list all locks in the currently connected database                      | ACTIVITY METRICS BASE                        |

*Table 878. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

*Table 879. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

This element can help explain what is happening when an application is waiting to obtain a lock on an object. While it may appear that the application already has a lock on the object it needs, it may have to wait to obtain a different type of lock on the same object.

The lock can be in one of the following statuses:

- G** Granted state: The application has the lock in the state specified by the **lock\_mode** monitor element.
- C** Converting state: The application is trying to change the lock held to a different type; for example, changing from a share lock to an exclusive lock.
- W** Waiting state.

**Note:** API users should refer to the sqlmon.h header file containing definitions of database system monitor constants.

---

## lock\_timeout\_val - Lock timeout value monitor element

Indicates the timeout value (in seconds) when an application has issued a SET CURRENT LOCK TIMEOUT statement. In cases where the statement has not been executed, the database level lock timeout will be shown.

Table 880. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | agent                 | Basic          |

Table 881. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | -                     | -              |

### Usage

The SET CURRENT LOCK TIMEOUT statement can be used to specify the maximum duration for which application agents will wait for a table or index lock.

If an application is waiting too long on a lock, you can check the **lock\_timeout\_val** monitor element value to see whether it is set too high inside the application. You can modify the application to lower the lock timeout value to let the application time out, if that is appropriate for the application logic. You can accomplish this modification with the SET CURRENT LOCK TIMEOUT statement.

If the application is timing out frequently, you can check whether the lock timeout value is set too low and increase it as appropriate.

---

## lock\_timeouts - Number of lock timeouts monitor element

The number of times that a request to lock an object timed out instead of being granted.

Table 882. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 882. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 883. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 884. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db   | Always collected      |
| Connection    | event_conn   | Always collected      |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

This element can help you adjust the setting for the **locktimeout** database configuration parameter. If the number of lock timeouts becomes excessive when compared to normal operating levels, you may have an application that is holding locks for long durations. In this case, this element may indicate that you should analyze some of the other lock and deadlock monitor elements to determine if you have an application problem.

You could also have too few lock timeouts if your **locktimeout** database configuration parameter is set too high. In this case, your applications may wait excessively to obtain a lock.

---

## lock\_timeouts\_global - Lock timeouts global monitor element

Number of lock timeouts where the application holding the lock was on a remote member.

*Table 885. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 886. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this element in conjunction with the **lock\_timeouts** monitor element. The **lock\_timeouts\_global** monitor element represents the number of times a lock timeout has occurred while waiting to acquire a lock held on another member. To determine the number of times a lock timeout has occurred while waiting to acquire a lock held on the same member, use the following formula:

`lock_timeouts - lock_timeouts_global`

Outside of the DB2 pureScale environment, this value is always zero.

---

## lock\_wait\_end\_time - Lock wait end timestamp monitor element

The date and time the application stopped waiting to obtain a lock on the object that is currently lock.

Table 887. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## lock\_wait\_start\_time - Lock wait start timestamp monitor element

The date and time that this application started waiting to obtain a lock on the object that is currently locked by another application.

Table 888. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

Table 889. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch  |
|----------------|-----------------------|-----------------|
| Application    | appl                  | Lock, Timestamp |
| Lock           | lock_wait             | Lock, Timestamp |

Table 890. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Timestamp        |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Timestamp        |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This element can help you determine the severity of resource contention.

## lock\_wait\_time - Time waited on locks monitor element

The total elapsed time spent waiting for locks. The value is given in milliseconds.

Table 891. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |

Table 891. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                              | DATA OBJECT METRICS | EXTENDED         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS     | BASE             |

Table 892. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Lock           |
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | appl_lock_list |

For snapshot monitoring, this counter can be reset.

Table 893. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Transactions  | event_xact  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

At the database level, this is the total amount of elapsed time that all applications were waiting for a lock within this database. This measure of elapsed time can include time spent on locks taken during activities, as well as locks taken during other processing, such compilation.

At the application-connection and transaction levels, this is the total amount of elapsed time that this connection or transaction has waited for a lock to be granted to it.

The value for this element does not include lock wait times for agents that are currently still in a lock wait state. It only includes lock wait times for agents that have already completed their lock waits.

This element may be used in conjunction with the **lock\_waits** monitor element to calculate the average wait time for a lock. This calculation can be performed at either the database or the application-connection level.

When using monitor elements providing elapsed times, you should consider:

- Elapsed times are affected by system load, so the more processes you have running, the higher this elapsed time value.
- To calculate this element at the database level, the database system monitor sums the application-level times. This can result in double counting elapsed times at a database level, since more than one application process can be running at the same time.

To provide meaningful data, you can calculate the average wait time for a lock, as previously shown.

---

## lock\_wait\_time\_global - Lock wait time global monitor element

Time spent on global lock waits. The unit of measurement for time is in milliseconds.

*Table 894. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 894. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element              | Collection Level |
|---|------------------------------|------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE         |                  |
| MON_GET_TABLE table function - get table metrics  | DATA OBJECT METRICS EXTENDED |                  |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list   | DATA OBJECT METRICS EXTENDED |                  |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE         |                  |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE         |                  |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE         |                  |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE         |                  |

Table 895. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element in conjunction with the **lock\_wait\_time** monitor element, which represents all the time spent waiting for locks. The **lock\_wait\_time\_global** monitor element represents the time spent waiting for locks held by conflicting applications on different members. To determine the total time spent waiting for locks held by conflicting applications on the same member, use the following formula:

$$\text{lock\_wait\_time} - \text{lock\_wait\_time\_global}$$

Outside of the DB2 pureScale environment, this value is always zero.

---

## lock\_wait\_time\_global\_top - Top global lock wait time monitor element

The longest lock wait that has occurred for a lock that is held on another member. This value is reported in milliseconds.

Table 896. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_wlstats         | Always collected |

---

## lock\_wait\_time\_top – Lock wait time top monitor element

The high watermark for lock wait times of any request in a workload. Units are milliseconds. The lock\_wait\_time\_top high watermark is always collected for workloads. A request contributes toward this high watermark only when request metrics are enabled.

Table 897. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_wlstats         | Always collected |

### Usage

Use this element to determine the highest lock wait time of any request on a partition for a workload during the time interval collected.

---

## lock\_wait\_val - Lock wait value monitor element

The amount of time spent in lock wait (in milliseconds) before an event for mon\_lockwait is generated.

Table 898. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## lock\_waits - Lock waits monitor element

The total number of times that applications or connections waited for locks.

Table 899. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 899. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 900. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 901. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

Table 901. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Database      | event_db                                   | Always collected      |
| Connection    | event_conn                                 | Always collected      |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

At the database level, this is the total number of times that applications have had to wait for locks within this database.

At the application-connection level, this is the total number of times that this connection requested a lock but had to wait because another connection was already holding a lock on the data.

This element may be used with **lock\_wait\_time** to calculate, at the database level, the average wait time for a lock. This calculation can be done at either the database or the application-connection level.

If the average lock wait time is high, you should look for applications that hold many locks, or have lock escalations, with a focus on tuning your applications to improve concurrency, if appropriate. If escalations are the reason for a high average lock wait time, then the values of one or both of the **locklist** and **maxlocks** configuration parameters may be too low.

---

## lock\_waits\_global - Lock waits global monitor element

Number of lock waits due to the application holding the lock being on a remote member.

Table 902. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 902. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element              | Collection Level |
|---|------------------------------|------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE         |                  |
| MON_GET_TABLE table function - get table metrics  | DATA OBJECT METRICS EXTENDED |                  |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list   | DATA OBJECT METRICS EXTENDED |                  |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE         |                  |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE         |                  |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE         |                  |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE         |                  |

Table 903. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element in conjunction with the **lock\_waits** monitor element, which reports the total number of lock waits due to locks held by conflicting applications on all members. The **lock\_waits\_global** monitor element indicates the number of times that a lock wait was held by conflicting applications on different members. To determine the number of lock waits held by a conflicting application on the same member as the waiting application, use the following formula:

$$\text{lock\_waits} - \text{lock\_waits\_global}$$

Outside of the DB2 pureScale environment, this value is always zero.

---

## locks\_held - Locks held monitor element

The number of locks currently held.

*Table 904. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 905. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |
| Lock           | db_lock_list          | Basic          |
| Lock           | appl_lock_list        | Basic          |

*Table 906. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks with Details | event_detailed_dlconn | Always collected |

### Usage

If the monitor information is at the database level, this is the total number of locks currently held by all applications in the database.

If the monitor information is at the application level, this is the total number of locks currently held by all agents for the application.

---

## locks\_held\_top - Maximum number of locks held monitor element

The maximum number of locks held during this transaction.

*Table 907. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Transactions | event_xact            | Always collected |

### Usage

You can use this element to determine if your application is approaching the maximum number of locks available to it, as defined by the **maxlocks** configuration parameter. This parameter indicates the percentage of the lock list that each

application can use before lock escalations occur. Lock escalations can result in a decrease in concurrency between applications connected to a database.

Since the **maxlocks** parameter is specified as a percentage and this element is a counter, you can compare the count provided by this element against the total number of locks that can be held by an application, as calculated using one of the following formulae:

- On 64-bit systems,  $(locklist * 4096 / 64) * (maxlocks / 100)$
- On 32-bit systems,  $(locklist * 4096 / 48) * (maxlocks / 100)$

If you have a large number of locks, you may need to perform more commits within your application so that some of the locks can be released.

---

## locks\_in\_list - Number of Locks Reported

The number of locks held by a particular application to be reported on by the event monitor.

*Table 908. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks with Details | event_detailed_dlconn | Always collected |

---

## locks\_waiting - Current agents waiting on locks monitor element

Indicates the number of agents waiting on a lock.

*Table 909. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |
| Lock           | db_lock_list          | Basic          |

### Usage

When used in conjunction with **appls\_cur\_cons**, this element indicates the percentage of applications waiting on locks. If this number is high, the applications may have concurrency problems, and you should identify applications that are holding locks or exclusive locks for long periods of time.

---

## log\_buffer\_wait\_time - Log buffer wait time monitor element

The amount of time an agent spends waiting for space in the log buffer. The value is given in milliseconds.

*Table 910. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 910. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 911. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## log\_disk\_wait\_time - Log disk wait time monitor element

The amount of time an agent spends waiting for log records to be flushed to disk. The value is given in milliseconds.

Table 912. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 913. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |

Table 913. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activitymetrics                                | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## log\_disk\_waits\_total - Total log disk waits monitor element

The number of times agents have to wait for log data to write to disk.

Table 914. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 914. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 915. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## log\_held\_by\_dirty\_pages - Amount of Log Space Accounted for by Dirty Pages

The amount of log (in bytes) corresponding to the difference between the oldest dirty page in the database and the top of the active log.

### Element identifier

log\_held\_by\_dirty\_pages

### Element type

watermark

Table 916. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

Table 917. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 918. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** When the snapshot is taken, this value is calculated based on conditions at the time of that snapshot.

Use this element to evaluate the effectiveness of page cleaning for older pages in the buffer pool.

The cleaning of old pages in the buffer pool is governed by the *softmax* database configuration parameter. If the page cleaning is effective then *log\_held\_by\_dirty\_pages* should be less than or approximately equal to:

$$(\text{softmax} / 100) * \text{logfilesiz} * 4096$$

If this statement is not true, increase the number of page cleaners (*num\_iocleaners*) configuration parameter.

If the condition is true and it is required that less log be held by dirty pages, then decrease the *softmax* configuration parameter.

## log\_read\_time - Log Read Time

The total elapsed time spent by the logger reading log data from the disk. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

Table 919. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 920. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 921. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *log\_reads*, *num\_log\_read\_io*, and *num\_log\_data\_found\_in\_buffer* elements to determine if:

- The current disk is adequate for logging.
- The log buffer size is adequate.

## log\_reads - Number of Log Pages Read

The number of log pages read from disk by the logger.

**Element identifier**

log\_reads

**Element type**

counter

Table 922. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 923. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 924. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You can use this element with an operating system monitor to quantify the amount of I/O on a device that is attributable to database activity.

---

## log\_to\_redo\_for\_recovery - Amount of Log to be Redone for Recovery

The amount of log (in bytes) that will have to be redone for crash recovery.

**Element identifier**

log\_to\_redo\_for\_recovery

**Element type**

watermark

Table 925. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 926. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 927. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** When the snapshot is taken, this value is calculated based on conditions at the time of that snapshot. Larger values indicate longer recovery times after a system crash. If the value seems excessive, check the *log\_held\_by\_dirty\_pages* monitor element to see if page cleaning needs to be tuned. Also check if there are any long running transactions that need to be terminated.

---

## log\_write\_time - Log Write Time

The total elapsed time spent by the logger writing log data to the disk. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

*Table 928. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 929. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 930. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *log\_writes* and *num\_log\_write\_io* elements to determine if the current disk is adequate for logging.

---

## log\_writes - Number of Log Pages Written

The number of log pages written to disk by the logger.

**Element identifier**

log\_writes

**Element type**

counter

*Table 931. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 932. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 933. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You may use this element with an operating system monitor to quantify the amount of I/O on a device that is attributable to database activity.

**Note:** When log pages are written to disk, the last page might not be full. In such cases, the partial log page remains in the log buffer, and additional log records are written to the page. Therefore log pages might be written to disk by the logger more than once. You should not use this element to measure the number of pages produced by DB2.

---

## long\_object\_pages - Long Object Pages

The number of disk pages consumed by long data in a table.

*Table 934. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 935. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

**Usage** This element provides a mechanism for viewing the actual amount of space consumed by long data in a particular table. This element can be used in conjunction with a table event monitor to track the rate of long data growth over time.

---

## long\_object\_l\_pages - Long object data logical pages monitor element

The number of logical pages used on disk by long data contained in this table.

*Table 936. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

### Usage

- This value might be less than the amount of space physically allocated for the object. This can happen when you use the REUSE STORAGE option of the TRUNCATE statement. This option causes storage allocated for the table to continue to be allocated, although the storage will be considered empty. In addition, the value for this monitor element might be less than the amount of space logically allocated for the object, because the total space logically allocated includes a small amount of additional meta data.

To retrieve an accurate measure of the logical or physical size of an object, use the ADMIN\_GET\_TAB\_INFO\_V97 function. This function provides more accurate information about the size of objects than you can obtain by multiplying the number of pages reported for this monitor element by the page size.

---

## long\_tbsp\_id - Long table space ID monitor element

An identifier of the table space that holds long data (LONG or LOB type columns) for this table.

*Table 937. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE                        |

### Usage

The value of this element matches a value from column TBSPACEID of view SYSCAT.TABLESPACES.

---

## machine\_identification - Host hardware identification monitor element

A string that describes the processor architecture. For example, "x86 64 bit". The value returned for this identifier is determined by the operating system running on the host.

*Table 938. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table function - Return system information | ACTIVITY METRICS BASE            |

---

## max\_agent\_overflows - Maximum Agent Overflows

The number of times a request to create a new agent was received when the Maximum Number of Agents (**maxagents**) configuration parameter had already been reached.

**Note:** The **max\_agent\_overflows** monitor element is deprecated starting with DB2 Version 9.5. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 939. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

If agent creation requests are still being received when the **maxagents** configuration parameter has been reached, this might indicate too high a workload for this node.

---

## max\_coord\_stmt\_exec\_time - Maximum coordinator statement execution time monitor element

The maximum coordinator execution time of a single run of the statement, in milliseconds. This value will be zero at non-coordinator nodes or if the statement has not been executed.

Table 940. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 941. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Package cache | pkgcache              | ACTIVITY METRICS BASE |

---

## max\_coord\_stmt\_exec\_time\_args - Maximum coordinator statement execution time arguments monitor element

An XML document that represents the input arguments provided to the statement that was running when the maximum execution time of a single run of the statement (**max\_coord\_stmt\_exec\_time**) took place on the coordinator member. This column is NULL if the statement has not yet run or if the statement has no input arguments.

This document contains a parent element named **max\_coord\_stmt\_exec\_time\_args**, which consists of one or more elements with the name **max\_coord\_stmt\_exec\_time\_arg**. See Figure 19 on page 944 for an example of the structure of the XML document.

Table 942. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 943. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Package cache | pkgcache_stmt_args    | ACTIVITY METRICS BASE |

## Usage

You can view the content of this document with the XMLPARSE scalar function. For example:

```
SELECT XMLPARSE(DOCUMENT MAX_COORD_STMT_EXEC_STMT_ARGS)
       FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL, NULL, NULL, -2));
```

Figure 19 shows an example of the contents of XML document returned by the preceding statement.

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<max_coord_stmt_exec_time_args
  xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="10010000">
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>1</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>INTEGER</stmt_value_type>
    <stmt_value_data>5</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>2</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>VARCHAR</stmt_value_type>
    <stmt_value_data>78</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>3</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>VARCHAR</stmt_value_type>
    <stmt_value_data>john</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>4</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>VARCHAR</stmt_value_type>
    <stmt_value_data>x</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>5</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>DATE</stmt_value_type>
    <stmt_value_data>2001-02-12</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
  :
  <max_coord_stmt_exec_time_arg>
    <stmt_value_index>15</stmt_value_index>
    <stmt_value_isreopt id="0">no</stmt_value_isreopt>
    <stmt_value_isnull id="0">no</stmt_value_isnull>
    <stmt_value_type>DECIMAL</stmt_value_type>
    <stmt_value_data>+0002000.55</stmt_value_data>
  </max_coord_stmt_exec_time_arg>
</max_coord_stmt_exec_time_args>
```

---

Figure 19. Sample content of `max_coord_stmt_exec_time_args`. In this example, the document shows that 15 arguments were passed to the statement.

Entries for the following data types are recorded in this XML document, however, the actual values for arguments of these types are not captured in the STMT\_VALUE\_DATA element:

- BLOB
- CLOB
- REF
- BOOLEAN
- Structured data types
- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML tyoes
- DBCLOB
- ARRAY types
- ROW types
- ROWID
- CURSOR variables

Input arguments are recorded beginning with the one that appears first in the statement, and continuing with each one in succession. The number of input parameters that can be recorded is constrained only by the upper limit on the size of the BLOB containing the XML document. In practical terms, it is unlikely the number input arguments captured would ever cause this limit to be reached.

The schema for the XML document that contains this element can be found in the path `sqllib/misc/DB2EvmonPkgCache.xsd`.

---

## **max\_coord\_stmt\_exec\_timestamp - Maximum coordinator statement execution timestamp monitor element**

The time the statement that produced the `max_coord_stmt_exec_time` value began execution.

*Table 944. Table Function Monitoring Information*

| <b>Table Function</b>   | <b>Monitor Element Collection Level</b> |
|---|---|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE                   |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE                   |

*Table 945. Event Monitoring Information*

| <b>Event Type</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-------------------|------------------------------|-----------------------|
| Package cache     | pkgcache                     | ACTIVITY METRICS BASE |

---

## **max\_data\_received\_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes**

This element represents the number of statements or chains with outbound bytes received between 513 and 1024 inclusive.

**Element identifier**  
max\_data\_received\_1024

**Element type**  
counter

*Table 946. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_received\_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes**

This element represents the number of statements or chains with outbound bytes received between 1 and 128 inclusive.

**Element identifier**  
max\_data\_received\_128

**Element type**  
counter

*Table 947. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_received\_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes**

This element represents the number of statements or chains with outbound bytes received between 8193 and 16384 inclusive.

**Element identifier**  
max\_data\_received\_16384

**Element type**  
counter

Table 948. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_received\_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes

This element represents the number of statements or chains with outbound bytes received between 1025 and 2048 inclusive.

**Element identifier**

max\_data\_received\_2048

**Element type**

counter

Table 949. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_received\_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes

This element represents the number of statements or chains with outbound bytes received between 129 and 256 inclusive.

**Element identifier**

max\_data\_received\_256

**Element type**

counter

Table 950. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_received\_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element**

This element represents the number of statements or chains with outbound bytes received between 16385 and 31999 inclusive.

*Table 951. Snapshot Monitoring Information*

| <b>Snapshot Level</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-----------------------|------------------------------|-----------------------|
| DCS Database          | dcс_dbase                    | Statement             |
| DCS Application       | dcс_appl                     | Statement             |
| Data Transmission     | stmt_transmissions           | Statement             |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_received\_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes**

This element represents the number of statements or chains with outbound bytes received between 2049 and 4096 inclusive.

**Element identifier**

max\_data\_received\_4096

**Element type**

counter

*Table 952. Snapshot Monitoring Information*

| <b>Snapshot Level</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-----------------------|------------------------------|-----------------------|
| DCS Database          | dcс_dbase                    | Statement             |
| DCS Application       | dcс_appl                     | Statement             |
| Data Transmission     | stmt_transmissions           | Statement             |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_received\_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes**

This element represents the number of statements or chains with outbound bytes received between 257 and 512 inclusive.

**Element identifier**

max\_data\_received\_512

**Element type**  
counter

*Table 953. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

### **max\_data\_received\_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element**

This element represents the number of statements or chains with outbound bytes received between 32000 and 64000 inclusive.

*Table 954. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

### **max\_data\_received\_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes**

This element represents the number of statements or chains with outbound bytes received between 4097 and 8192 inclusive.

**Element identifier**  
max\_data\_received\_8192

**Element type**  
counter

*Table 955. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_received\_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes

This element represents the number of statements or chains with outbound bytes received greater than 64000.

**Element identifier**

max\_data\_received\_gt64000

**Element type**

counter

*Table 956. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes

This element represents the number of statements or chains with outbound bytes sent between 513 and 1024 inclusive.

**Element identifier**

max\_data\_sent\_1024

**Element type**

counter

*Table 957. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes

This element represents the number of statements or chains with outbound bytes sent between 1 and 128 inclusive.

**Element identifier**  
max\_data\_sent\_128

**Element type**  
counter

*Table 958. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_sent\_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes**

This element represents the number of statements or chains with outbound bytes sent between 8193 and 16384 inclusive.

**Element identifier**  
max\_data\_sent\_16384

**Element type**  
counter

*Table 959. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_data\_sent\_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes**

This element represents the number of statements or chains with outbound bytes sent between 1025 and 2048 inclusive.

**Element identifier**  
max\_data\_sent\_2048

**Element type**  
counter

Table 960. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes

This element represents the number of statements or chains with outbound bytes sent between 129 and 256 inclusive.

**Element identifier**

max\_data\_sent\_256

**Element type**

counter

Table 961. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes

This element represents the number of statements or chains with outbound bytes sent between 16385 and 31999 inclusive.

**Element identifier**

max\_data\_sent\_31999

**Element type**

counter

Table 962. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes

This element represents the number of statements or chains with outbound bytes sent between 2049 and 4096 inclusive.

**Element identifier**

max\_data\_sent\_4096

**Element type**

counter

*Table 963. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes

This element represents the number of statements or chains with outbound bytes sent between 257 and 512 inclusive.

**Element identifier**

max\_data\_sent\_512

**Element type**

counter

*Table 964. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes

This element represents the number of statements or chains with outbound bytes sent between 32000 and 64000 inclusive.

**Element identifier**

max\_data\_sent\_64000

**Element type**

counter

*Table 965. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes

This element represents the number of statements or chains with outbound bytes sent between 4097 and 8192 inclusive.

**Element identifier**

max\_data\_sent\_8192

**Element type**

counter

*Table 966. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Statement      |
| DCS Application   | dcs_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_data\_sent\_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes

This element represents the number of statements or chains with outbound bytes sent greater than 64000.

**Element identifier**

max\_data\_sent\_gt64000

**Element type**  
counter

*Table 967. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_network\_time\_100\_ms - Number of Statements with Network Time between 16 and 100 ms**

This element represents the number of statements or chains whose network time was greater than 16 milliseconds but less or equal to 100 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

*Table 968. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## **max\_network\_time\_16\_ms - Number of Statements with Network Time between 4 and 16 ms**

This element represents the number of statements or chains whose network time was greater than 4 milliseconds but less or equal to 16 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

**Element identifier**  
max\_network\_time\_16\_ms

**Element type**  
counter

*Table 969. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Statement      |
| DCS Application   | dc_s_appl             | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_network\_time\_1\_ms - Number of Statements with Network Time of up to 1 ms

This element represents the number of statements or chains whose network time was less or equal to 1 millisecond. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

*Table 970. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_network\_time\_4\_ms - Number of Statements with Network Time between 1 and 4 ms

This element represents the number of statements or chains whose network time was greater than 1 millisecond but less or equal to 4 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

**Element identifier**

max\_network\_time\_4\_ms

**Element type**

counter

*Table 971. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_network\_time\_500\_ms - Number of Statements with Network Time between 100 and 500 ms

This element represents the number of statements or chains whose network time was greater than 100 milliseconds but less or equal to 500 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

*Table 972. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## max\_network\_time\_gt500\_ms - Number of Statements with Network Time greater than 500 ms

This element represents the number of statements or chains whose network time was greater than 500 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

*Table 973. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Statement      |
| DCS Application   | dcс_appl              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to get a better idea of the database activity and network traffic at the database or application levels.

---

## member - Database member monitor element

The numeric identifier for the database member from which the data was retrieved for this result record.

*Table 974. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_MEM_USAGE table function - Get total memory consumption for instance | ACTIVITY METRICS BASE            |
| AUDIT_ARCHIVE procedure and table function - Archive audit log file            | ACTIVITY METRICS BASE            |

Table 974. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| DBCFCG administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information | ACTIVITY METRICS BASE            |
| ENV_GET_REG_VARIABLES table function - Retrieve DB2 registry settings in use                                     | ACTIVITY METRICS BASE            |
| ENV_GET_DB2_SYSTEM_RESOURCES table function - Return DB2(r) system information                                   | ACTIVITY METRICS BASE            |
| ENV_GET_SYSTEM_RESOURCES table function - Return system information  | ACTIVITY METRICS BASE            |
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information                                    | ACTIVITY METRICS BASE            |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details  | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs                          | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation            | ACTIVITY METRICS BASE            |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | ACTIVITY METRICS BASE            |
| MON_GET_CF_WAIT_TIME table function - Get cluster caching facility command wait time                             | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics   | ACTIVITY METRICS BASE            |
| MON_GET_EXTENDED_LATCH_WAIT table function - Return information for latches                                      | ACTIVITY METRICS BASE            |
| MON_GET_EXTENT_MOVEMENT_STATUS - Get extent movement progress status metrics                                     | ACTIVITY METRICS BASE            |
| MON_GET_FCM - Get FCM metrics  | ACTIVITY METRICS BASE            |
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections  | ACTIVITY METRICS BASE            |
| MON_GET_GROUP_BUFFERPOOL table function - Get group buffer pool metrics  | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics   | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list                           | ACTIVITY METRICS BASE            |

Table 974. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_LOCKS table function - list all locks in the currently connected database               | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information                                | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function - Get memory set information                                  | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information              | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space              | ACTIVITY METRICS BASE            |
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests      | ACTIVITY METRICS BASE            |
| MON_GET_SERVERLIST table function - get member priority details                                 | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics         | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list           | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics                                     | ACTIVITY METRICS BASE            |
| MON_GET_TRANSACTION_LOG table function - Get log information                                    | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                 | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list                   | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD table function - Get workload metrics  | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                         | ACTIVITY METRICS BASE            |

Table 974. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element  | Collection Level |
|---|------------------|------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics  | ACTIVITY METRICS | BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample   | ACTIVITY METRICS | BASE             |
| PD_GET_DIAG_HIST table function - Return<br>records from a given facility   | ACTIVITY METRICS | BASE             |
| PDLOGMSGs_LAST24HOURS<br>administrative view and<br>PD_GET_LOG_MSGS table function -<br>Retrieve problem determination messages | ACTIVITY METRICS | BASE             |
| WLM_GET_QUEUE_STATS table function -<br>Return threshold queue statistics   | ACTIVITY METRICS | BASE             |
| WLM_GET_SERVICE_CLASS_AGENTS table<br>function - list agents running in a service<br>class                                      | ACTIVITY METRICS | BASE             |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list<br>workload occurrences                                    | ACTIVITY METRICS | BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service<br>subclasses                                   | ACTIVITY METRICS | BASE             |
| WLM_GET_SERVICE_SUPERCLASS_STATS<br>table function - Return statistics of service<br>superclasses                               | ACTIVITY METRICS | BASE             |
| WLM_GET_WORK_ACTION_SET_STATS<br>table function - Return work action set<br>statistics  | ACTIVITY METRICS | BASE             |
| WLM_GET_WORKLOAD_OCCURRENCE<br>_ACTIVITIES table function - Return a list of<br>activities                                      | ACTIVITY METRICS | BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics   | ACTIVITY METRICS | BASE             |

Table 975. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | -  | COLLECT BASE DATA     |
| Locking       | -  | Always collected      |

Table 975. Event Monitoring Information (continued)

| Event Type     | Logical Data Grouping   | Monitor Switch   |
|----------------|---|------------------|
| Change history | changesummary<br>dbdbmcfg<br>regvar<br>ddlstmexec<br>txncompletion<br>evmonstart<br>utilstart<br>utillocation<br>utilstop | Always collected |

## Usage

A DB2 member is a database manager instance that runs DB2 server software on a single host. A DB2 member accepts and processes database requests from applications connected to it.

---

## memory\_free - Amount of free physical memory monitor element

The total amount of physical memory on this host that is not allocated to a running process, in MB.

Table 976. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table function - Return system information | ACTIVITY METRICS BASE            |

---

## memory\_pool\_used\_hwm - Memory pool high water mark monitor element

The highest amount of memory assigned to this pool since it was created, in KB.

Table 977. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_POOL table function - Get memory set information | ACTIVITY METRICS BASE            |

---

## memory\_pool\_id - Memory pool identifier monitor element

Memory pool identifier

Table 978. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_MEMORY_POOL table function - Get memory pool information | Always collected                 |

## memory\_pool\_type - Memory pool name monitor element

The name of the memory pool.

Table 979. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_POOL table function -<br>Get memory pool information | ACTIVITY METRICS BASE            |

### Usage

Use the memory\_pool\_type element to identify the type of memory pool. The possible values returned by this monitor element are listed in Table 980.

Table 980. Possible values returned for memory\_pool\_type.

| Memory pool name* | Description                                   | Additional information  |
|-------------------|---|---|
| APM               | Agent pool management (APM) heap              | Internal memory pool  |
| APPL_SHARED       | Application shared heap                       | Internal memory pool  |
| APPLICATION       | Application heap                              | See applheapsz - Application heap size configuration parameter.       |
| APS               | APS heap                                      | Internal memory pool  |
| BSU_CF            | Base service utility (BSU) CF heap            | Internal memory pool  |
| BSU               | Base service utility (BSU) heap               | Internal memory pool  |
| BP                | Buffer pool heap                              | See CREATE BUFFERPOOL statement.                                      |
| CAT_CACHE         | Catalog cache heap                            | See catalogcache_sz - Catalog cache size configuration parameter.     |
| DATABASE_CF       | Database CF heap                              | Internal memory pool  |
| DATABASE          | Database heap                                 | See dbheap - Database heap configuration parameter.                   |
| DEBUG             | Debug heap                                    | Internal memory pool  |
| DROP_INDEX        | Drop index heap                               | Internal memory pool  |
| EDU               | Engine dispatchable unit (EDU) heap           | Internal memory pool  |
| FCMBP             | Fast communications manager (FCM) buffer heap | See fcm_num_buffers - Number of FCM buffers configuration parameter.  |
| FCM_CHANNEL       | FCM channel heap                              | See fcm_num_channels - Number of FCM channels configuration parameter |
| FCM_CONTROL       | FCM control heap                              | Internal memory pool  |
| FCM_LOCAL         | FCM local heap                                | Internal memory pool  |
| FCM_SESSION       | FCM session heap                              | Internal memory pool  |
| FEDERATED         | Federated heap                                | Internal memory pool  |

Table 980. Possible values returned for `memory_pool_type` (continued).

| Memory pool name*  | Description                | Additional information   |
|--------------------|----------------------------|--|
| KERNEL_CONTROL     | Kernel control block heap  | Internal memory pool   |
| KERNEL             | Kernel heap                | Internal memory pool   |
| LOCK_MGR           | Lock manager heap          | See locklist - Maximum storage for lock list configuration parameter.              |
| MISC               | Miscellaneous heap         | See DB2_FMP_COMM_HEAPSZ registry variable.   |
| MONITOR            | Monitor heap               | See mon_heap_sz - Database system monitor heap size configuration parameter.       |
| OPTPROF_PARSER     | OptProf XML parser heap    | Internal memory pool   |
| OSS_TRACKER        | OSS resource tracking heap | Internal memory pool   |
| PERSISTENT_PRIVATE | Persistent private heap    | Internal memory pool   |
| PACKAGE_CACHE      | Package cache heap         | See pckcachesz - Package cache size configuration parameter.                       |
| PRIVATE            | Private                    | Internal memory pool   |
| RESYNC             | Resync heap                | Internal memory pool   |
| SORT               | Private sort heap          | See sortheap - Sort heap size configuration parameter.                             |
| SHARED_SORT        | Shared sort heap           | See sheapthres_shr - Sort heap threshold for shared sorts configuration parameter. |
| SQL_COMPILER       | SQL compiler heap          | Internal memory pool   |
| STATEMENT          | Statement heap             | See stmtheap - Statement heap size configuration parameter.                        |
| STATISTICS         | Statistics heap            | See stat_heap_sz - Statistics heap size configuration parameter.                   |
| USER_DATA          | User data heap             | Internal memory pool   |
| UTILITY            | Utility heap               | See util_heap_sz - Utility heap size configuration parameter.                      |
| XMLCACHE           | XML cache heap             | Internal memory pool   |
| XMLPARSER          | XML parser heap            | Internal memory pool   |

\* Names returned by `db2pd` for these pools might be abbreviated. The names of the pools and any abbreviations used by `db2pd` are defined in `sqlpoolinfo.h`.

## memory\_pool\_used - Amount of memory pool in use monitor element

Amount of committed memory in use by this memory pool, in KB.

Table 981. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_POOL table function -<br>Get memory pool information | ACTIVITY METRICS BASE            |

---

## memory\_set\_committed - Memory currently committed monitor element

The amount of memory currently committed to this memory set, in KB.

*Table 982. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_SET table function -<br>Get memory set information | ACTIVITY METRICS BASE            |

### Usage

Committed memory is memory that is backed by RAM, or paging space, or both on the system.

---

## memory\_set\_id - Memory set identifier monitor element

A numeric identifier that maps to a specific memory set type.

*Table 983. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_SET table function -<br>Get memory set information | ACTIVITY METRICS BASE            |

---

## memory\_set\_size - Memory set size monitor element

Maximum memory commitment limit, in KB.

This value represents either a configured setting for a memory set or an internally calculated value for those memory sets that are managed automatically.

*Table 984. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_SET table function -<br>Get memory set information | ACTIVITY METRICS BASE            |

---

## memory\_set\_type - Memory set type monitor element

The type of memory set.

*Table 985. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_MEMORY_POOL table function -<br>Get memory pool information | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function -<br>Get memory set information   | ACTIVITY METRICS BASE            |

### Usage

The possible values returned for this monitor element are described in Table 986 on page 965:

Table 986. Possible values for `memory_set_type`

| Memory set type | Description                                 | Scope          |
|-----------------|---|----------------|
| DBMS            | Database manager memory set                 | Instance       |
| FMP             | Fenced mode process memory set              | Instance       |
| PRIVATE         | Private memory set                          | Instance       |
| DATABASE        | Database memory set                         | Database       |
| APPLICATION     | Application memory set                      | Database       |
| FCM             | Fast communication manager (FCM) memory set | Instance, Host |

---

## memory\_set\_used - Memory in use by this set monitor element

Amount of memory from this set that has been assigned to memory pools, in KB.

Table 987. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_MEMORY_SET table function - Get memory set information | ACTIVITY METRICS BASE            |

### Usage

All of memory represented by this monitor element is committed memory; the value returned for this monitor element is included in `MEMORY_SET_COMMITTED`. Any additional memory that is committed but not in use is cached to improve performance.

---

## memory\_set\_used\_hwm - Memory set high water mark monitor element

The highest amount of memory assigned to memory pools from this set since the memory set was created, in KB.

Table 988. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_MEMORY_SET table function - Get memory set information | ACTIVITY METRICS BASE            |

---

## memory\_swap\_free - Total free swap space monitor element

The total amount of unused swap space on this host, in MB.

Table 989. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table function - Return system information | ACTIVITY METRICS BASE            |

---

## memory\_swap\_total - Total swap space monitor element

The total amount of swap space on this host, in MB.

*Table 990. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## memory\_total - Total physical memory monitor element

The total amount of physical memory on this host, in MB.

*Table 991. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## message - Control Table Message

The nature of the timestamp in the MESSAGE\_TIME column. This element is only used in the CONTROL table by write-to-table event monitors.

*Table 992. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| -          | -                     | Always collected |

### Usage

The following possible values are:

**DROPPED RECORDS: *n***

Number of activity records that were dropped because MONHEAP could not be allocated for them.

**FIRST\_CONNECT**

The time of the first connect to the database after activation.

**EVMON\_START**

The time the event monitor listed in the EVMONNAME column was started.

**OVERFLOWS: *n***

Denotes that *n* records were discarded due to buffer overflow.

**LAST DROPPED RECORD**

The last time that an activity record was dropped.

---

## message\_time - Timestamp Control Table Message

The timestamp corresponding to the event described in the MESSAGE column. This element is only used in the CONTROL table by write-to-table event monitors.

Table 993. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| -          | -                     | Always collected |

---

## mon\_interval\_id - Monitor interval identifier monitor element

The value of the MON\_INTERVAL\_ID database global variable when a particular transaction is completed.

Table 994. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list  | ACTIVITY METRICS BASE            |

Table 995. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Statistics   | event_qstats          | Always collected |
| Statistics   | event_scstats         | Always collected |
| Statistics   | event_histogrambin    | Always collected |
| Statistics   | event_wcstats         | Always collected |
| Statistics   | event_wlstats         | Always collected |
| Activities   | event_activity        | Always collected |
| Unit of Work | uow                   | Always collected |

---

## nesting\_level - Nesting level monitor element

This element shows the level of nesting or recursion in effect when the statement was being run; each level of nesting corresponds to nested or recursive invocation of a stored procedure or user-defined function (UDF).

The **nesting\_level** monitor element is an alias of the **stmt\_nest\_level** monitor element.

Table 996. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 997. Event Monitoring Information

| Event Type   | Logical Data Grouping         | Monitor Switch |
|--|-------------------------------|----------------|
| Locking  | -                             | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history            | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history            | -              |
| Activities   | event_activitystmt            | -              |
| Unit of work                                       | Reported in the package list. | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element, along with **stmt\_invocation\_id** monitor element, to uniquely identify the invocation in which a particular SQL statement has been executed. You can also use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

---

## network\_time\_bottom - Minimum Network Time for Statement

This element represents the shortest network time for a statement executed against this DCS database or in this DCS application, or having used this many data transmissions. (Network time is the difference between host response time and elapsed execution time for a statement.)

### Element identifier

network\_time\_bottom

### Element type

watermark

Table 998. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dcs_dbase             | Statement, Timestamp |
| DCS Application   | dcs_appl              | Statement, Timestamp |
| Data Transmission | stmt_transmissions    | Statement, Timestamp |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to get a better idea of the database activity and network traffic at the database or application levels.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3

and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## network\_time\_top - Maximum Network Time for Statement

This element represents the longest network time for a statement executed against this DCS database or in this DCS application, or having used this many data transmissions. (Network time is the difference between host response time and elapsed execution time for a statement.)

**Element identifier**

network\_time\_top

**Element type**

watermark

*Table 999. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dcs_dbase             | Statement, Timestamp |
| DCS Application   | dcs_appl              | Statement, Timestamp |
| Data Transmission | stmt_transmissions    | Statement, Timestamp |

For snapshot monitoring, this counter can be reset.

### Usage

Use this element to get a better idea of the database activity and network traffic at the database or application levels. Note that this element is not collected when the timestamp switch is off.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## nleaf - Number of leaf pages monitor element

The approximate number of leaf pages.

*Table 1000. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## nlevels - Number of index levels monitor element

Number of index levels. This is an approximation.

*Table 1001. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## no\_change\_updates - Number of no change row updates monitor element

The number of row updates that resulted in no changes to any of the column values of a row. If a LOB, XML, or LONG column is included in the SET clause of an UPDATE statement, every row affected by that statement is assumed to change, and will not participate in this counter.

*Table 1002. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - get table metrics | ACTIVITY METRICS BASE            |

---

## node\_number - Node Number

The number assigned to the node in the *db2nodes.cfg* file.

*Table 1003. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |
| Database Manager | memory_pool           | Basic          |
| Database Manager | fcm                   | Basic          |
| Database Manager | fcm_node              | Basic          |
| Database Manager | utility_info          | Basic          |
| Database         | detail_log            | Basic          |
| Buffer Pool      | bufferpool_nodeinfo   | Buffer Pool    |
| Table Space      | rollforward           | Basic          |
| Lock             | lock                  | Basic          |
| Lock             | lock_wait             | Basic          |
| Database         | db_sto_path_info      | Buffer Pool    |

*Table 1004. Event Monitoring Information*

| Event Type      | Logical Data Grouping | Monitor Switch   |
|-----------------|-----------------------|------------------|
| Connections     | event_connheader      | Always collected |
| Deadlocks       | lock                  | Always collected |
| Overflow Record | event_overflow        | Always collected |
| Database        | event_dbmemuse        | Always collected |
| Connection      | event_connmemuse      | Always collected |

**Usage** This value identifies the current node number, which can be used when monitoring multiple nodes.

---

## nonboundary\_leaf\_node\_splits - Non-boundary leaf node splits monitor element

Number of times a non-boundary leaf node was split during an insert operation.

*Table 1005. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## num\_agents - Number of Agents Working on a Statement

Number of concurrent agents currently executing a statement or subsection.

**Element identifier**

num\_agents

**Element type**

gauge

*Table 1006. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| Application    | subsection            | Statement      |

**Usage** An indicator how well the query is parallelized. This is useful for tracking the progress of query execution, by taking successive snapshots.

---

## num\_assoc\_agents - Number of Associated Agents

At the application level, this is the number of subagents associated with an application. At the database level, it is the number of subagents for all applications.

**Element identifier**

num\_assoc\_agents

**Element type**

gauge

*Table 1007. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl_info             | Basic          |

**Usage** You can use this element to help evaluate your settings for your agent configuration parameters.

---

## num\_compilations - Statement Compilations

The number of different compilations for a specific SQL statement.

Table 1008. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

**Usage** Some SQL statements issued on different schemas, such as **SELECT t1 FROM test** will appear to be the same statement in the DB2 cache even though they refer to different access plans. Use this value in conjunction with num\_executions to determine whether a bad compilation environment may be skewing the results of dynamic SQL snapshot statistics.

---

## num\_coord\_exec - Number of executions by coordinator agent monitor element

The number of times this section was executed by a coordinator agent.

Table 1009. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1010. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

---

## num\_coord\_exec\_with\_metrics - Number of executions by coordinator agent with metrics monitor element

The number of times this section was executed by a coordinator agent and monitoring metrics were being captured

Table 1011. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1012. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

---

## num\_db\_storage\_paths - Number of automatic storage paths

The number of automatic storage paths associated with a database.

Table 1013. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

### Usage note

You can use this element with the `db_storage_path` monitor element to identify the storage paths that are associated with this database.

If you use storage groups, this element shows the number of automatic storage paths in the default database storage group only.

---

## num\_executions - Statement executions monitor element

The number of times that an SQL statement has been executed.

Table 1014. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1015. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1016. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

### Usage

You can use this element to identify the most frequently executed SQL statements in your system.

At the package cache level, use this element to compute averages for the activity metrics reported per statement. For example, the average CPU usage for an execution of a statement reported at the package cache level can be calculated by the following formula:

`total_cpu_time / num_exec_with_metrics`

Use the `num_exec_with_metrics` monitor element instead of the `num_executions` monitor element when computing averages, since the `num_executions` monitor element counts all executions of a statement, regardless of whether or not the

execution of the statement contributed to the activity metrics that are reported.

---

## num\_exec\_with\_metrics - Number of executions with metrics collected monitor element

The number of times that this SQL statement section has been executed with the metrics collected. This element can be used to calculate the per execution value for monitor elements for statements in the package cache.

Table 1017. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1018. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

---

## num\_extents\_left - Number of extents left to process monitor element

The number of extents left to move during this table rebalancing process.

Table 1019. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_EXTENT_MOVEMENT_STATUS - Get extent movement progress status metrics | ACTIVITY METRICS BASE            |

---

## num\_extents\_moved - Number of extents moved monitor element

The number of extents moved so far during this extent movement operation.

Table 1020. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_EXTENT_MOVEMENT_STATUS - Get extent movement progress status metrics | ACTIVITY METRICS BASE            |

---

## num\_gw\_conn\_switches - Connection Switches

The number of times that an agent from the agents pool was primed with a connection and was reassigned for use with a different DRDA database.

Table 1021. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

## Usage

For most users, the default setting of the **num\_poolagents** configuration parameter ensures optimal performance. The default setting for this configuration parameter automatically manages agent pooling and avoids reassigning agents.

To reduce the value of this monitor element, adjust the value of the **num\_poolagents** configuration parameter.

---

## num\_indoubt\_trans - Number of Indoubt Transactions

The number of outstanding indoubt transactions in the database.

*Table 1022. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

*Table 1023. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** Indoubt transactions hold log space for uncommitted transactions, which can cause the logs to become full. When the logs are full, further transactions cannot be completed. The resolution of this problem involves a manual process of heuristically resolving the indoubt transactions. This monitor element provides a count of the number of currently outstanding indoubt transactions that must be heuristically resolved.

---

## num\_log\_buffer\_full - Number of times full log buffer caused agents to wait monitor element

The number of times agents have to wait for log data to write to disk while copying log records into the log buffer. This value is incremented per agent per incident. For example, if two agents attempt to copy log data while the buffer is full, then this value is incremented by two.

*Table 1024. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1024. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TRANSACTION_LOG table function - Get log information   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1025. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1026. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element to determine if the `logbufsz` database configuration parameter needs to be increased.

---

## num\_log\_data\_found\_in\_buffer - Number of Log Data Found In Buffer

The number of times an agent reads log data from the buffer. Reading log data from the buffer is preferable to reading from the disk because the latter is slower.

*Table 1027. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1028. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1029. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the `num_log_read_io` element to determine if the `LOGBUFSZ` database configuration parameter needs to be increased.

---

## num\_log\_part\_page\_io - Number of Partial Log Page Writes

The number of I/O requests issued by the logger for writing partial log data to the disk.

*Table 1030. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1031. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1032. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *log\_writes*, *log\_write\_time*, and *num\_log\_write\_io* elements to determine if the current disk is adequate for logging.

---

## num\_log\_read\_io - Number of Log Reads

The number of I/O requests issued by the logger for reading log data from the disk.

*Table 1033. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1034. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1035. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *log\_reads* and *log\_read\_time* elements to determine if the current disk is adequate for logging.

---

## num\_log\_write\_io - Number of Log Writes

The number of I/O requests issued by the logger for writing log data to the disk.

*Table 1036. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1037. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1038. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** Use this element in conjunction with the *log\_writes* and *log\_write\_time* elements to determine if the current disk is adequate for logging.

## num\_lw\_thresh\_exceeded - Number of lock wait thresholds exceeded monitor element

This monitor element reports the number of times the lock wait threshold (set using **mon\_lw\_thresh** configuration parameter) was exceeded and a lock wait event was captured by the locking event monitor. If no lock wait event is generated, the monitor element is not incremented.

*Table 1039. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1040. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 1040. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## num\_nodes\_in\_db2\_instance - Number of Nodes in Partition

The number of nodes on the instance where the snapshot was taken.

Table 1041. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

Table 1042. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |

**Usage** Use this element to determine the number of nodes for an instance. For non-partitioned system databases, this value will be 1.

## num\_page\_dict\_built - Number of page-level compression dictionaries created or recreated

The number of page-level compression dictionaries created or recreated for a table since the database was last activated.

Table 1043. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - get table metrics | ACTIVITY METRICS BASE            |

## num\_ref\_with\_metrics - Number of references with metrics monitor element

The total number of times that a section referenced the database object. The usage list for the data object must be created and active; in addition the collection of object metrics must be enabled for the section.

Table 1044. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | ACTIVITY METRICS BASE            |

Table 1044. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list | ACTIVITY METRICS BASE            |

---

## num\_references - Number of references monitor element

The number of times that this section has referenced this object since it was added to the list.

Table 1045. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list  | ACTIVITY METRICS BASE            |

---

## num\_remaps - Number of remaps monitor element

Count of the number of times this activity has been remapped. If num\_remaps is greater than zero, the service\_class\_id of this activity record is the ID of the last service class to which the activity was remapped.

Table 1046. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this information to verify whether the activity was remapped the expected number of times.

---

## num\_tbsps - Number of table spaces monitor element

The number of table spaces associated with a logged event.

Table 1047. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change history | utilstart             | Always collected |

---

## num\_threshold\_violations - Number of threshold violations monitor element

The number of threshold violations that have taken place in this database since it was last activated.

This monitor element is an alias of the “thresh\_violations - Number of threshold violations monitor element” on page 1304 monitor element, which is returned by some monitoring (MON\_\*) table functions.

*Table 1048. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1049. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

## Usage

This element can be used to help determine whether or not thresholds are effective for this particular application or whether the threshold violations are excessive.

---

## num\_transmissions - Number of Transmissions

Number of data transmissions between the DB2 Connect gateway and the host that was used to process this DCS statement. (One data transmission consists of either one send or one receive.)

### Note:

This is a legacy monitor element that is not relevant for DB2 UDB Version 8.1.2 or higher. If you are using DB2 UDB Version 8.1.2 or higher, refer to the **num\_transmissions\_group** monitor element.

### Element identifier

num\_transmissions

### Element type

counter

-->

*Table 1050. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Statement  | dcs_stmt              | Statement      |

**Usage** Use this element to get a better understanding of the reasons why a particular statement took longer to execute. For example, a query returning a large result set may need many data transmissions to complete.

---

## num\_transmissions\_group - Number of Transmissions Group

The range of data transmissions between the DB2 Connect gateway and the host that was used to process this DCS statement. (One data transmission consists of either one send or one receive.)

Table 1051. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Statement  | dcs_stmt              | Statement      |

**Usage** Use this element to get a better understanding of the reasons why a particular statement took longer to execute. For example, a query returning a large result set may need many data transmissions to complete.

The constants representing the ranges of transmissions are described as follows and are defined in sqlmon.h.

| API Constant                | Description                   |
|-----------------------------|-------------------------------|
| SQLM_DCS_TRANS_GROUP_2      | 2 transmissions               |
| SQLM_DCS_TRANS_GROUP_3TO7   | 3 to 7 transmissions          |
| SQLM_DCS_TRANS_GROUP_8TO15  | 8 to 15 transmissions         |
| SQLM_DCS_TRANS_GROUP_16TO64 | 16 to 64 transmissions        |
| SQLM_DCS_TRANS_GROUP_GT64   | Greater than 64 transmissions |

---

## number\_in\_bin - Number in bin monitor element

This element holds the count of the number of activities or requests that fall within the histogram bin.

Table 1052. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

### Usage

Use this element to represent the height of a bin in the histogram.

---

## object\_data\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent data pages found in local buffer pool monitor element

The number of group buffer pool (GBP) independent data pages found in a local buffer pool (LBP) by an agent.

Table 1053. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

Table 1054. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

## object\_data\_gbp\_invalid\_pages - GBP invalid data pages for a table monitor element

The number of times that a data page is requested from the group buffer pool (GBP) for a table. The page is requested because the version of the page in the local buffer pool (LBP) is invalid. Outside of a DB2 pureScale environment, this value is null.

Table 1055. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

To determine how often a requested data page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\frac{(\text{object\_data\_lbp\_pages\_found} + \text{object\_xda\_lbp\_pages\_found})}{(\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads})}$$

To determine how often a requested data page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$\frac{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads} - \text{object\_data\_GBP\_p\_reads} - \text{object\_xda\_p\_reads})}{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads})}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

## object\_data\_gbp\_l\_reads - GBP data logical reads for a table monitor element

The number of times that a group buffer pool (GBP) dependent data page is requested from the GBP for a table. The page is requested because a valid version of the page does not exist in the local buffer pool (LBP).

Table 1056. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | DATA OBJECT METRICS EXTENDED     |

Table 1056. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested data page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\frac{(\text{object\_data\_lbp\_pages\_found} + \text{object\_xda\_lbp\_pages\_found})}{(\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads})}$$

To determine how often a requested data page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$\frac{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads} - \text{object\_data\_GBP\_p\_reads} - \text{object\_xda\_p\_reads})}{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads})}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_data\_gbp\_p\_reads - GBP data physical reads for a table monitor element

The number of times that a group buffer pool (GBP) dependent data page is read into the local buffer pool (LBP) from disk for a table. The page is read from disk into the LBP because the page is not in the GBP.

Table 1057. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested data page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\frac{(\text{object\_data\_lbp\_pages\_found} + \text{object\_xda\_lbp\_pages\_found})}{(\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads})}$$

To determine how often a requested data page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$\frac{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads} - \text{object\_data\_GBP\_p\_reads} - \text{object\_xda\_p\_reads})}{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads})}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or

GBP might be limiting the throughput of your database.

---

## object\_data\_lbp\_pages\_found - LBP data pages found for a table monitor element

The number of times that a data page for a table is present in the local buffer pool (LBP).

*Table 1058. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

To determine how often a requested data page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\frac{(\text{object\_data\_lbp\_pages\_found} + \text{object\_xda\_lbp\_pages\_found})}{(\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads})}$$

To determine how often a requested data page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$\frac{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads} - \text{object\_data\_GBP\_p\_reads} - \text{object\_xda\_p\_reads})}{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads})}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_data\_l\_reads - Buffer pool data logical reads for a table monitor element

The number of data pages that are logically read from the buffer pool for a table.

*Table 1059. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

This monitor element tracks the number of accesses to the following data:

- Data that is in the buffer pool when the database manager needs to process the page

- Data that is read into the buffer pool before the database manager can process the page

Calculate the data page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_data\_p\_reads} + \text{object\_xda\_p\_reads}) / (\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

## object\_data\_p\_reads - Buffer pool physical data reads for a table monitor element

The number of data pages that are physically read for a table.

*Table 1060. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

Calculate the data page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_data\_p\_reads} + \text{object\_xda\_p\_reads}) / (\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

## object\_index\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent index pages found in local buffer pool monitor element

The number of Group Buffer Pool (GBP) independent index pages found in Local Buffer Pool (LBP) by agent.

*Table 1061. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

*Table 1062. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |

Table 1062. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping | Monitor Switch       |
|--------------|-----------------------|----------------------|
| Unit of Work | uow_metrics           | REQUEST METRICS BASE |

## object\_index\_gbp\_invalid\_pages - GBP invalid index pages for an index monitor element

The number of times that an index page is requested from the group buffer pool (GBP) for an index. The page is requested because the version of the page in the local buffer pool (LBP) is invalid.

Table 1063. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

To determine how often a requested index page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\text{object\_index\_lbp\_pages\_found} / \text{object\_index\_l\_reads}$$

To determine how many times a requested index page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$(\text{object\_index\_gbp\_l\_reads} - \text{object\_index\_gbp\_p\_reads}) / \text{object\_index\_gbp\_l\_reads}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

## object\_index\_gbp\_l\_reads - GBP index logical reads for an index monitor element

The number of times that a group buffer pool (GBP) dependent index page is requested from the GBP for an index. The page is requested because a valid version of the page does not exist in the local buffer pool (LBP).

Table 1064. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested index page is found in the LBP, use the following formula, which uses the values of monitor elements:

**object\_index\_lbp\_pages\_found / object\_index\_l\_reads**

To determine how many times a requested index page is found in the GBP, use the following formula, which also uses the values of monitor elements:

**(object\_index\_gbp\_l\_reads - object\_index\_gbp\_p\_reads) / object\_index\_gbp\_l\_reads**

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_index\_gbp\_p\_reads - GBP index physical reads for an index monitor element

The number of times that a group buffer pool (GBP) dependent index page is read into the local buffer pool (LBP) from disk for an index. The page is read from disk into the LBP because the page is not in the GBP.

*Table 1065. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested index page is found in the LBP, use the following formula, which uses the values of monitor elements:

**object\_index\_lbp\_pages\_found / object\_index\_l\_reads**

To determine how many times a requested index page is found in the GBP, use the following formula, which also uses the values of monitor elements:

**(object\_index\_gbp\_l\_reads - object\_index\_gbp\_p\_reads) / object\_index\_gbp\_l\_reads**

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_index\_lbp\_pages\_found - LBP index pages found for an index monitor element

The number of times that an index page for an index is present in the local buffer pool (LBP).

*Table 1066. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | DATA OBJECT METRICS EXTENDED     |

Table 1066. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested index page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\text{object\_index\_lbp\_pages\_found} / \text{object\_index\_l\_reads}$$

To determine how many times a requested index page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$(\text{object\_index\_gbp\_l\_reads} - \text{object\_index\_gbp\_p\_reads}) / \text{object\_index\_gbp\_l\_reads}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

## object\_index\_l\_reads - Buffer pool index logical reads for an index monitor element

The number of index pages that are logically read from the buffer pool for an index.

Table 1067. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

This monitor element tracks the number of accesses to the following pages:

- Index pages that are in the buffer pool when the database manager needs to process the pages
- Index pages that are read into the buffer pool before the database manager can process the pages

Calculate the index page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_index\_p\_reads} / \text{object\_index\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

---

## object\_index\_p\_reads - Buffer pool index physical reads for an index

The number of index pages that are physically read for an index.

*Table 1068. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics                                       | DATA OBJECT METRICS EXTENDED     |
| MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

Calculate the index page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_index\_p\_reads} / \text{object\_index\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

---

## object\_name - Object name monitor element

A table name or an index name, depending on the value of the **objtype** monitor element.

*Table 1069. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_AUTO_MAINT_QUEUE table function - Get information about automatic maintenance jobs            | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation | ACTIVITY METRICS BASE            |
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests            | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status of a usage list                         | ACTIVITY METRICS BASE            |

*Table 1070. Event Monitoring Information*

| Event Type     | Logical Data Grouping  | Monitor Switch   |
|----------------|------------------------|------------------|
| Change history | utilphase<br>utilstart | Always collected |

### Usage

For the change history event monitor, if the **object\_type** element is INDEX, PARTIONGROUP, or TABLE, then this is the name of the index, partition group, or table.

---

## object\_requested - Requested object monitor element

The type of lock the requester is trying to acquire from the owner. Values can be LOCK for database locks, or TICKET for WLM tickets..

Table 1071. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## object\_schema - Object schema monitor element

A table schema or an index schema, depending on the value of the **objtype** monitor element.

Table 1072. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_AUTO_MAINT_QUEUE table function - Get information about automatic maintenance jobs            | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation | ACTIVITY METRICS BASE            |
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests            | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status of a usage list                         | ACTIVITY METRICS BASE            |

Table 1073. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Switch   |
|----------------|------------------------|------------------|
| Change history | utilphase<br>utilstart | Always collected |

### Usage

For the change history event monitor, if the **object\_type** element is INDEX or TABLE, then this is the schema of the index or table, otherwise it is an empty string.

---

## object\_xda\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element

The number of group buffer pool (GBP) independent XML storage object (XDA) data pages found in a local buffer pool (LBP) by an agent.

Table 1074. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - get table metrics | DATA OBJECT METRICS EXTENDED     |

Table 1074. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

Table 1075. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

## object\_xda\_gbp\_invalid\_pages - GBP invalid XDA data pages for a table monitor element

The number of times that a data page for an XML storage object (XDA) is requested from the group buffer pool (GBP) for a table. The page is requested because the version of the page in the local buffer pool (LBP) is invalid.

Table 1076. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

To determine how often a requested XDA page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\text{object\_xda\_lbp\_pages\_found} / \text{object\_xda\_l\_reads}$$

To determine how many times a requested XDA page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$(\text{object\_xda\_gbp\_l\_reads} - \text{object\_xda\_gbp\_p\_reads}) / \text{object\_xda\_gbp\_l\_reads}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

## object\_xda\_gbp\_l\_reads - GBP XDA data logical read requests for a table monitor element

The number of times that a group buffer pool (GBP) dependent data page for an XML storage object (XDA) is requested from the GBP for a table. The page is requested because a valid version of the page does not exist in the local buffer pool (LBP).

Table 1077. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested XDA page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\text{object\_xda\_lbp\_pages\_found} / \text{object\_xda\_l\_reads}$$

To determine how many times a requested XDA page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$(\text{object\_xda\_gbp\_l\_reads} - \text{object\_xda\_gbp\_p\_reads}) / \text{object\_xda\_gbp\_l\_reads}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_xda\_gbp\_p\_reads - GBP XDA data physical read requests for a table monitor element

The number of times that a group buffer pool (GBP) dependent data page for an XML storage object (XDA) is read into the local buffer pool (LBP) from disk for a table. The page is read from disk into the LBP because the page is not in the GBP.

Table 1078. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

To determine how often a requested XDA page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\text{object\_xda\_lbp\_pages\_found} / \text{object\_xda\_l\_reads}$$

To determine how many times a requested XDA page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$(\text{object\_xda\_gbp\_l\_reads} - \text{object\_xda\_gbp\_p\_reads}) / \text{object\_xda\_gbp\_l\_reads}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_xda\_lbp\_pages\_found - LBP XDA data pages found for a table monitor element

The number of times that an XML storage object (XDA) data page for a table is present in the local buffer pool (LBP).

*Table 1079. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

To determine how often a requested data page is found in the LBP, use the following formula, which uses the values of monitor elements:

$$\frac{(\text{object\_data\_lbp\_pages\_found} + \text{object\_xda\_lbp\_pages\_found})}{(\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads})}$$

To determine how often a requested data page is found in the GBP, use the following formula, which also uses the values of monitor elements:

$$\frac{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads} - \text{object\_data\_GBP\_p\_reads} - \text{object\_xda\_p\_reads})}{(\text{object\_data\_GBP\_l\_reads} + \text{object\_xda\_l\_reads})}$$

LBP and GBP hit rates are important factors in the overall performance of a cluster caching facility. Using these formulas can help you determine whether the LBP or GBP might be limiting the throughput of your database.

---

## object\_xda\_l\_reads - Buffer pool XDA data logical reads for a table monitor element

The number of data pages for XML storage objects (XDAs) that are logically read from the buffer pool for a table.

*Table 1080. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

This monitor element tracks the number of accesses to the following data:

- Data that is in the buffer pool when the database manager needs to process the page.
- Data that is read into the buffer pool before the database manager can process the page

Calculate the data page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_data\_p\_reads} + \text{object\_xda\_p\_reads}) / (\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

## object\_xda\_p\_reads - Buffer pool XDA data physical reads for a table monitor element

The number of data pages for XML storage objects (XDAs) that are physically read for a table.

*Table 1081. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | DATA OBJECT METRICS EXTENDED     |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

### Usage

Calculate the data page hit ratio by using the following formula, which uses the values of monitor elements:

$$(1 - (\text{object\_data\_p\_reads} + \text{object\_xda\_p\_reads}) / (\text{object\_data\_l\_reads} + \text{object\_xda\_l\_reads}))$$

If the hit ratio is low, increasing the number of buffer pool pages might improve performance.

## objtype - Object type monitor element

The type of object for which monitoring data is being reported. This monitor element is an alias of the object\_type monitor element.

*Table 1082. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings                             | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries       | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs               | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information                    | ACTIVITY METRICS BASE            |

Table 1082. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list              | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility                     | ACTIVITY METRICS BASE            |

Table 1083. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Switch   |
|----------------|------------------------|------------------|
| Change history | utilphase<br>utilstart | Always collected |

### Usage Notes

- When this element is returned by the ADMIN\_GET\_TAB\_COMPRESS\_INFO table function or the ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function, the returned value for the object\_type monitor element can be either 'XML' or 'DATA'.
- When this element is returned by the MON\_GET\_RTS\_RQST table function, the returned value for the object\_type monitor element is 'TABLE'.
- When this element is returned by the MON\_GET\_AUTO\_MAINT\_QUEUE table function, the returned value for the object\_type monitor element can be 'DATABASE', 'TABLE', 'NICKNAME', or 'VIEW'.
- When this element is returned by the MON\_GET\_AUTO\_RUNSTATS\_QUEUE table function, the returned value for the object\_type monitor element can be 'TABLE', 'NICKNAME', or 'VIEW'.
- When this element is returned by the change history event monitor, the returned value for the object\_type monitor element can be 'DATABASE', 'INDEX', 'PARTITIONGROUP', 'TABLE' or 'TABLESPACE'.

## olap\_func\_overflows - OLAP Function Overflows monitor element

The number of times that OLAP function data exceeded the available sort heap space.

Table 1084. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1085. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

## Usage

At the database level, use this element in conjunction with `total_olap_funcs` to calculate the percentage of OLAP functions that overflowed to disk. If this percentage is high and the performance of applications using OLAP functions needs to be improved, then you should consider increasing the sort heap size.

At the application level, use this element to evaluate OLAP function performance for individual applications.

---

## open\_cursors - Number of Open Cursors

The number of cursors currently open for an application.

Table 1086. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl              | Statement      |

**Usage** Use this element to assess how much memory is being allocated. The amount of memory allocated by the DB2 client, DB2 Connect, or the database agent on the target database is related to the number of cursors that are currently open. Knowing this information can help with capacity planning. For example, each open cursor that is doing blocking has a buffer size of `RQRIOBLK`. If `deferred_prepare` is enabled, then two buffers will be allocated.

This element does not include cursors that were closed by an early close. An early close occurs when the host database returns the last record to the client. The cursor is closed at the host and gateway, but is still open at the client. Early close cursors can be set using the DB2 Call Level Interface.

---

## open\_loc\_curs - Open Local Cursors

The number of local cursors currently open for this application, including those cursors counted by `open_loc_curs_blk`.

**Element identifier**

`open_loc_curs`

**Element type**

gauge

Table 1087. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

**Usage** You may use this element in conjunction with `open_loc_curs_blk` to calculate the percentage of local cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application.

For cursors used by remote applications, see `open_rem_curs`.

---

## open\_loc\_curs\_blk - Open Local Cursors with Blocking

The number of local blocking cursors currently open for this application.

**Element identifier**  
open\_loc\_curs\_blk

**Element type**  
gauge

Table 1088. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

**Usage** You may use this element in conjunction with *open\_loc\_curs* to calculate the percentage of local cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application:

- Check the pre-compile options for record blocking for treatment of ambiguous cursors
- Redefine cursors to allow for blocking (for example, if possible, specify FOR FETCH ONLY on your cursors).

*rej\_curs\_blk* and *acc\_curs\_blk* provide additional information that may help you tune your configuration parameters to improve row blocking in your application.

For blocking cursors used by remote applications, see *open\_rem\_curs\_blk*.

---

## open\_rem\_curs - Open Remote Cursors

The number of remote cursors currently open for this application, including those cursors counted by *open\_rem\_curs\_blk*.

**Element identifier**  
open\_rem\_curs

**Element type**  
gauge

Table 1089. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

**Usage** You may use this element in conjunction with *open\_rem\_curs\_blk* to calculate the percentage of remote cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application. See *open\_rem\_curs\_blk* for more information.

For the number of open cursors used by applications connected to a local database, see *open\_loc\_curs*.

---

## open\_rem\_curs\_blk - Open Remote Cursors with Blocking

The number of remote blocking cursors currently open for this application.

**Element identifier**  
open\_rem\_curs\_blk

**Element type**  
gauge

Table 1090. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

**Usage** You can use this element in conjunction with *open\_rem\_curs* to calculate the percentage of remote cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application:

- Check the pre-compile options for record blocking for treatment of ambiguous cursors
- Redefine cursors to allow for blocking (for example, if possible, specify FOR FETCH ONLY on your cursors).

*rej\_curs\_blk* and *acc\_curs\_blk* provide additional information that may help you tune your configuration parameters to improve row blocking in your application.

For the number of open blocking cursors used by applications connected to a local database see *open\_loc\_curs\_blk*.

---

## os\_level - Operating system level monitor element

The modification level of the operating system running on this host. Reported for Linux systems only.

Table 1091. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## os\_name - Operating system name monitor element

The name of the operating system running on this host.

Table 1092. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## os\_release - Operating system release monitor element

The release of the operating system running on this host.

Table 1093. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## os\_version - Operating system version monitor element

The version of the operating system running on this host.

*Table 1094. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## outbound\_appl\_id - Outbound Application ID

This identifier is generated when the application connects to the DRDA host database. It is used to connect the DB2 Connect gateway to the host, while the **appl\_id** monitor element is used to connect a client to the DB2 Connect gateway.

### Element identifier

outbound\_appl\_id

### Element type

information

*Table 1095. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

## Usage

You may use this element in conjunction with **appl\_id** to correlate the client and server parts of the application information.

This identifier is unique across the network.

This element will be blank when the gateway concentrator is on, or if the DCS application is not in a logical unit of work.

### Format

Network.LU Name.Application instance

### Example

CAIBMTOR.OSFDBM0.930131194520

---

## outbound\_bytes\_received - Outbound Number of Bytes Received

The number of bytes received by the DB2 Connect gateway from the host, excluding communication protocol usage (for example, TCP/IP). For the data transmission level: Number of bytes received by the DB2 Connect gateway from the host during the processing of all the statements that used this number of data transmissions.

*Table 1096. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Database    | dcs_dbase             | Basic          |
| DCS Application | dcs_appl              | Basic          |
| DCS Statement   | dcs_stmt              | Statement      |

Table 1096. Snapshot Monitoring Information (continued)

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring at the statement level, this counter cannot be reset. This counter can be reset at other levels.

### Usage

Use this element to measure the throughput from the host databases to the DB2 Connect gateway.

---

## outbound\_bytes\_received\_bottom - Minimum Outbound Number of Bytes Received

The lowest number of bytes received per statement or chain by the DB2 Connect gateway from the host during the processing of all the statements or chains against this DCS database, or in this DCS application, that used this number of data transmissions.

Table 1097. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Statement      |

**Usage** Use this element in conjunction with "outbound number of bytes received" as yet another parameter that illustrates the throughput from the host database to the DB2 Connect gateway.

---

## outbound\_bytes\_received\_top - Maximum Outbound Number of Bytes Received

Maximum number of bytes received per statement or chain by the DB2 Connect gateway from the host during the processing of all the statements or chains against this DCS database, or in this DCS application, that used this number of data transmissions.

Table 1098. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Statement      |

**Usage** Use this element in conjunction with "outbound number of bytes received" as yet another parameter that illustrates the throughput from the host database to the DB2 Connect gateway.

---

## outbound\_bytes\_sent - Outbound Number of Bytes Sent

The number of bytes sent by the DB2 Connect gateway to the host, excluding communication protocol usage (for example, TCP/IP). For the data transmission level: Number of bytes sent by the DB2 Connect gateway to the host during the processing of all the statements that used this number of data transmissions.

Table 1099. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcx_dbase             | Basic          |
| DCS Application   | dcx_appl              | Basic          |
| DCS Statement     | dcx_stmt              | Statement      |
| Data Transmission | stmt_transmissions    | Statement      |

For snapshot monitoring at the statement level, this counter cannot be reset. This counter can be reset at other levels.

**Usage** Use this element to measure the throughput from the DB2 Connect gateway to the host database.

---

## outbound\_bytes\_sent\_bottom - Minimum Outbound Number of Bytes Sent

The lowest number of bytes sent per statement or chain by the DB2 Connect gateway to the host during the processing of all the statements or chains against this DCS database, or in this DCS application, that used this number of data transmissions.

Table 1100. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Statement      |

**Usage** Use this element in conjunction with "outbound number of bytes sent" as yet another parameter that illustrates the throughput from the DB2 Connect Gateway to the host database.

---

## outbound\_bytes\_sent\_top - Maximum Outbound Number of Bytes Sent

Maximum number of bytes sent per statement or chain by the DB2 Connect gateway to the host during the processing of all the statements or chains against this DCS database, or in this DCS application, that used this number of data transmissions.

Table 1101. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Statement      |

**Usage** Use this element in conjunction with "outbound number of bytes sent" as yet another parameter that illustrates the throughput from the DB2 Connect Gateway to the host database.

---

## outbound\_comm\_address - Outbound Communication Address

This is the communication address of the target database. For example, it could be an IP address and port number for TCP/IP.

**Element identifier**  
outbound\_comm\_address

**Element type**  
information

*Table 1102. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

**Usage** Use this element for problem determination on DCS applications.

---

## **outbound\_comm\_protocol - Outbound Communication Protocol**

The communication protocol used between the DB2 Connect gateway and the host.

*Table 1103. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

### **Usage**

Use this element for problem determination on DCS applications. The valid value is:

- SQLM\_PROT\_TCPIP

---

## **outbound\_sequence\_no - Outbound Sequence Number**

This element will be blank when the gateway concentrator is on, or if the DCS application is not in a logical unit of work.

*Table 1104. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl_info         | Basic          |

---

## **overflow\_accesses - Accesses to overflowed records monitor element**

The number of accesses (reads and writes) to overflowed rows of this table.

*Table 1105. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

*Table 1106. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1107. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

## Usage

Overflowed rows indicate that data fragmentation has occurred. If this number is high, you may be able to improve table performance by reorganizing the table using the **REORG** utility, which cleans up this fragmentation.

A row overflows if it is updated and no longer fits in the data page where it was originally written. This usually happens as a result of an update of a VARCHAR or an ALTER TABLE statement.

---

## overflow\_creates - Overflow creates monitor element

The number of overflowed rows created on this table.

Table 1108. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

## Usage

---

## package\_id - Package identifier monitor element

A unique identifier for the package.

Table 1109. Event Monitoring Information

| Event Type   | Logical Data Grouping         | Monitor Switch   |
|--------------|-------------------------------|------------------|
| Unit of work | Reported in the package list. | Always collected |

## Usage

The value of this element matches a value from column PKGID of view SYSCAT.PACKAGES.

---

## package\_elapsed\_time - Package elapsed time monitor element

The elapsed time spent executing sections within the package. Value is in milliseconds.

Table 1110. Event Monitoring Information

| Event Type   | Logical Data Grouping         | Monitor Switch   |
|--------------|-------------------------------|------------------|
| Unit of work | Reported in the package list. | Always collected |

---

## package\_list\_count - Package list count monitor element

The number of entries that are present within the package listing for a particular unit of work

Table 1111. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |

---

## package\_list\_exceeded - Package list exceeded monitor element

Indicates whether the number of packages used within the unit of work has exceeded the capacity of the package list. Possible values are YES and NO.

Table 1112. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |

---

## package\_list\_size - Size of package list monitor element

The count of the number of package identifiers included in the package list.

Table 1113. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Unit of Work | uow                   |                |

---

## package\_name - Package name monitor element

The name of the package that contains the SQL statement.

Table 1114. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |

Table 1115. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

Table 1116. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch    |
|-------------------------------------|-----------------------|-------------------|
| Locking                             | -                     | -                 |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | -                 |
| Statements                          | event_stmt            | -                 |
| Activities                          | event_activystmt      | -                 |
| Package cache                       | -                     | COLLECT BASE DATA |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You may use this element to help identify the application program and the SQL statement that is executing.

---

## package\_schema - Package schema monitor element

The schema name of the package associated with an SQL statement.

Table 1117. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |

Table 1118. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Locking       | -                     | -                 |
| Package cache | -                     | COLLECT BASE DATA |

---

## package\_version\_id - Package version monitor element

For a given package name and creator, there can exist (starting in DB2 Version 8) multiple versions. The package version identifies the version identifier of the package that contains the SQL statement currently executing. The version of a package is determined at precompile (PREP) of the embedded SQL program using the VERSION keyword. If not specified at precompile time the package version has a value of "" (empty string).

Table 1119. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |

Table 1120. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 1121. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Locking       | -                     | Always collected  |
| Statements    | event_stmt            | Always collected  |
| Activities    | event_activitystmt    | Always collected  |
| Package cache | -                     | COLLECT BASE DATA |

## Usage

Use this element to help identify the package and the SQL statement that is currently executing.

---

## packet\_receive\_errors - Packet receive errors monitor element

Number of errors receiving packets since the network adapter started.

Table 1122. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

## packets\_received - Packets received monitor element

Number of packets received since the network adapter started.

Table 1123. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

## packet\_send\_errors - Packet send errors monitor element

Number of errors sending packets since the network adapter started.

*Table 1124. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

---

## packets\_sent - Packets sent monitor element

Number of packets sent since the network adapter started.

*Table 1125. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

---

## page\_allocations - Page allocations monitor element

Number of pages that have been allocated to the index.

*Table 1126. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

---

## page\_reorgs - Page reorganizations monitor element

The number of page reorganizations executed for a table.

*Table 1127. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | DATA OBJECT METRICS EXTENDED     |

---

*Table 1128. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

---

For snapshot monitoring, this counter can be reset.

*Table 1129. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

---

## Usage

Although a page might have enough space, the page could become fragmented in the following situations:

- When a new row is inserted
- When an existing row is updated, and the update results in an increased record size

A page might require reorganization when it becomes fragmented. Reorganization moves all fragmented space to a contiguous area, where the new record can be written. Such a page reorganization (page reorg) might require thousands of instructions. It also generates a log record of the operation.

Too many page reorganizations can result in less than optimal insert performance. You can use the REORG TABLE utility to reorganize a table and eliminate fragmentation. You can also use the APPEND parameter for the ALTER TABLE statement to indicate that all inserts are appended at the end of a table to avoid page reorganizations.

In situations where updates to rows causes the row length to increase, the page may have enough space to accommodate the new row, but a page reorg may be required to defragment that space. If the page does not have enough space for the new larger row, an overflow record is created causing *overflow\_accesses* during reads. You can avoid both situations by using fixed length columns instead of varying length columns.

---

## page\_reclaims\_x - Page reclaims exclusive access monitor element

The number of times a page related to the object was reclaimed by another member in the DB2 pureScale instance before its planned release, where the member that reclaimed the page required exclusive access.

*Table 1130. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

Page reclaims related to internally maintained object space maps are counted separately.

---

## page\_reclaims\_s - Page reclaims shared access monitor element

The number of times a page related to the object was reclaimed by another member in the DB2 pureScale instance before its planned release, where the member that reclaimed the page required shared access.

*Table 1131. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

Page reclaims related to internally maintained object space maps are counted separately.

---

### page\_reclaims\_initiated\_x - Page reclaims initiated exclusive access monitor element

The number of times a page accessed in exclusive mode caused the page to be reclaimed from another member. Page reclaims related to internally maintained object space maps are counted separately.

*Table 1132. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

---

### page\_reclaims\_initiated\_s - Page reclaims initiated shared access monitor element

The number of times a page accessed in shared mode caused the page to be reclaimed from another member. Page reclaims related to internally maintained object space maps are counted separately.

*Table 1133. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

---

### pages\_from\_block\_ios - Total number of pages read by block I/O monitor element

The total number of pages read by block I/O into the block area of the buffer pool.

*Table 1134. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics          | DATA OBJECT METRICS BASE                     |
| MON_GET_CONTAINER table function - Get table space container metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics          | DATA OBJECT METRICS BASE                     |

*Table 1135. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

## Usage

If block-based buffer pool is enabled, this element reports the total number of pages read by block I/O. Otherwise, this element returns 0.

To calculate the average number of pages sequentially prefetched per block-based I/O, divide the value of the **pages\_from\_block\_ios** monitor element by the value of the **block\_ios** monitor element. If this value is much less than the BLOCKSIZE option you have defined for the block-based buffer pool in the CREATE BUFFERPOOL or ALTER BUFFERPOOL statement, then block-based I/O is not being used to its full advantage. One possible cause for this is a mismatch between the extent size for the table space being sequentially prefetched and the block size of the block-based buffer pool.

---

## pages\_from\_vectored\_ios - Total number of pages read by vectored I/O monitor element

The total number of pages read by vectored I/O into the page area of the buffer pool.

*Table 1136. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics          | DATA OBJECT METRICS BASE                     |
| MON_GET_CONTAINER table function - Get table space container metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics          | DATA OBJECT METRICS BASE                     |

*Table 1137. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

---

## pages\_merged - Pages merged monitor element

Number of index pages that have been merged.

*Table 1138. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## pages\_read - Number of pages read monitor element

The number of pages (data, index, and XML) read in from the physical table space containers for regular and large table spaces.

*Table 1139. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONTAINER table function - Get table space container metrics | DATA OBJECT METRICS BASE         |

### Usage

---

## pages\_written - Number of pages written monitor element

The number of pages (data, index, and XML) physically written to the table space container.

*Table 1140. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONTAINER table function - Get table space container metrics | DATA OBJECT METRICS BASE         |

### Usage

---

## parent\_activity\_id - Parent activity ID monitor element

The unique ID of the activity's parent activity within the parent activity's unit of work. If there is no parent activity, the value of this monitor element is 0.

*Table 1141. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                       | ACTIVITY METRICS BASE                        |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE                        |

*Table 1142. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this element along with the **parent\_uow\_id** element and **app1\_id** element to uniquely identify the parent activity of the activity described in this activity record.

---

## parent\_uow\_id - Parent unit of work ID monitor element

The unique unit of work identifier within an application handle. The ID of the unit of work in which the activity's parent activity originates. If there is no parent activity, the value is 0.

Table 1143. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                       | ACTIVITY METRICS BASE                        |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE                        |

Table 1144. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this element along with the **parent\_activity\_id** element and **appl\_id** element to uniquely identify the parent activity of the activity described in this activity record.

---

## partial\_record - Partial Record monitor element

Indicates that an event monitor record is only a partial record.

Table 1145. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Database     | event_db              | -              |
| Tables       | event_table           | -              |
| Tablespaces  | event_tablespace      | -              |
| Bufferpools  | event_bufferpool      | -              |
| Connection   | event_conn            | -              |
| Statements   | event_stmt            | -              |
| Statements   | event_subsection      | -              |
| Transactions | event_xact            | -              |
| Activities   | event_activity        | -              |

### Usage

Most event monitors do not output their results until database deactivation. You can use the `FLUSH EVENT MONITOR <monitorName>` statement to force monitor values to the event monitor output writer. This allows you to force event monitor

records to the writer without needing to stop and restart the event monitor. This element indicates whether an event monitor record was the result of flush operation and so is a partial record.

Flushing an event monitor does not cause its values to be reset. This means that a complete event monitor record is still generated when the event monitor is triggered.

At the event\_activity logical data grouping, the possible values of **partial\_record** monitor element are:

- 0 The activity record was generated normally at the end of activity.
- 1 The activity record was generated as a result of calling the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS stored procedure.
- 2 Information is missing for this activity because not enough storage was available to create the records. Information may be missing from the event\_activity, event\_activystmt, or event\_activityvals records.

## participant\_no - Participant within Deadlock

A sequence number uniquely identifying this participant within this deadlock.

**Element identifier**  
participant\_no

**Element type**  
information

*Table 1146. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** Use this in your monitoring application to correlate deadlock connection event records with deadlock event records.

## participant\_no\_holding\_lk - Participant Holding a Lock on the Object Required by Application

The participant number of the application that is holding a lock on the object that this application is waiting to obtain.

**Element identifier**  
participant\_no\_holding\_lk

**Element type**  
information

*Table 1147. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** This element can help you determine which applications are in contention for resources.

---

## participant\_type - Participant type monitor element

The type of lock participant which can be either Requestor or Owner.

Table 1148. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## partition\_key - Partitioning key monitor element

The partitioning key for the event monitor tables. This value is chosen so that each event monitor process inserts data into the database partition on which the process is running; that is, insert operations are performed locally on the database partition where the event monitor process is running.

Table 1149. Event Monitoring Information

| Event Type          | Logical Data Grouping       | Monitor Switch |
|---------------------|-----------------------------|----------------|
| Threshold Violation | event_thresholdviolations   |                |
| Threshold Violation | control                     |                |
| Statistics          | event_qstats                |                |
| Statistics          | event_scstats               |                |
| Statistics          | event_histogrambin          |                |
| Statistics          | event_wcstats               |                |
| Statistics          | event_wcstats               |                |
| Statistics          | control                     |                |
| Locking             | lock                        |                |
| Locking             | lock_participants           |                |
| Locking             | lock_participant_activities |                |
| Locking             | lock_activity_values        |                |
| Locking             | control                     |                |
| Package Cache       | pkgcache_metrics            |                |
| Package Cache       | pkgcache_stmt_args          |                |
| Package Cache       | control                     |                |
| Unit of Work        | uow                         |                |
| Unit of Work        | uow_metrics                 |                |
| Unit of Work        | uow_package_list            |                |
| Unit of Work        | uow_executable_list         |                |
| Unit of Work        | control                     |                |
| Activities          | event_activity              |                |
| Activities          | event_activitystmt          |                |
| Activities          | event_activityvals          |                |
| Activities          | event_activitymetrics       |                |
| Activities          | control                     |                |

---

## partition\_number - Partition Number

This element is only used in the target SQL tables by write-to-table event monitors in a partitioned database environment or DB2 pureScale environment. This value indicates the number of the member where event monitor data is inserted.

*Table 1150. Table Function Monitoring Information*

| Table Function            | Monitor Element Collection Level |
|---------------------------|----------------------------------|
| DB_MEMBERS table function | ACTIVITY METRICS BASE            |

*Table 1151. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch        |
|------------|-----------------------|-----------------------|
| Activities | event_activitymetrics | ACTIVITY METRICS BASE |
| -          | -                     | Always collected      |

---

## passthru\_time - Pass-Through Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to PASSTHRU statements from all applications or a single application running on this federated server instance since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest. The response time is measured as the difference between the time the federated server submits a PASSTHRU statement to the data source, and the time it takes the data source to respond, indicating that the statement has been processed.

*Table 1152. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

### Usage

Use this element to determine how much actual time is spent at this data source processing statements in pass-through mode.

---

## passthru\_s - Pass-Through

This element contains a count of the total number of SQL statements that the federated server has passed through directly to this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

*Table 1153. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine what percentage of your SQL statements can be handled natively by the federated server, and what percentage requires pass-through mode. If this value is high, you should determine the cause and investigate ways to better use the native support.

---

## past\_activities\_wrapped - Past activities list wrapped monitor element

Indicates whether the activities list has wrapped. The default limit on the number of past activities to be kept by any one application is 250. This default can be overridden using the `DB2_MAX_INACT_STMTS` registry variable.. Users may want to choose a different value for the limit to increase or reduce the amount of system monitor heap used for inactive statement information.

Table 1154. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## phase\_start\_event\_id - Phase start event ID

The `EVENT_ID` of corresponding `UTILPHASESTART`.

Table 1155. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILPHASE             | Always collected |

### Usage

For the change history event monitor:

- If the event is the stopping of a utility phase or processing stage (`UTILPHASESTOP`), this is the `EVENT_ID` of the corresponding start of the utility phase (`UTILPHASESTART`), otherwise -1.

Use with the `PHASE_START_EVENT_TIMESTAMP` and member elements to associate the phase stop record with the corresponding start record.

---

## phase\_start\_event\_timestamp - Phase start event timestamp

Time of the corresponding `UTILPHASESTART`

Table 1156. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILPHASE             | Always collected |

### Usage

For the change history event monitor:

- If the event is the stopping of a utility phase or processing stage (`UTILPHASESTOP`), this is the time of the corresponding start of the utility phase (`UTILPHASESTART`), otherwise empty.

Use with the PHASE\_START\_EVENT\_ID and member elements to associate the phase stop record with the corresponding start record.

---

## pipedsortsaccepted - Piped Sorts Accepted

The number of piped sorts that have been accepted.

Table 1157. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

When the number of accepted piped sorts is low compared to the number requested, you can improve sort performance by adjusting one or both of the following configuration parameters:

- `sortheap`
- `sheapthres`

If piped sorts are being rejected, you might consider decreasing your sort heap or increasing your sort heap threshold. You should be aware of the possible implications of either of these options. If you increase the sort heap threshold, then there is the possibility that more memory will remain allocated for sorting. This could cause the paging of memory to disk. If you decrease the sort heap, you might require an extra merge phase that could slow down the sort.

---

## pipedsortsrequested - Piped Sorts Requested

The number of piped sorts that have been requested.

Table 1158. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

The sort list heap (`sortheap`) and sort heap threshold (`sheapthres`) configuration parameters help to control the amount of memory used for sort operations. These parameters are also used to determine whether a sort will be piped.

Since piped sorts may reduce disk I/O, allowing more piped sorts can improve the performance of sort operations and possibly the performance of the overall system. A piped sort is not accepted if the sort heap threshold will be exceeded when the sort heap is allocated for the sort. See *pipedsortsaccepted* for more information if you are experiencing piped sort rejections.

The SQL EXPLAIN output will show whether the optimizer requests a piped sort.

## pkg\_cache\_inserts - Package cache inserts monitor element

The total number of times that a requested section was not available for use and had to be loaded into the package cache. This count includes any implicit prepares performed by the system.

*Table 1159. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1160. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1161. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch       |
|------------|---|----------------------|
| Database   | event_db  | Always collected     |
| Connection | event_conn  | Always collected     |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics | event_wlstats (reported in the details_xml) document) | REQUEST METRICS BASE |

Table 1161. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                    | Monitor Switch       |
|--------------|--|----------------------|
| Unit of work | Reported in the system_metrics document. | REQUEST METRICS BASE |

## Usage

In conjunction with the **pkg\_cache\_lookups** monitor element, use this monitor element to calculate the package cache hit ratio using the following formula:

$$1 - (\text{Package Cache Inserts} / \text{Package Cache Lookups})$$

---

## pkg\_cache\_lookups - Package cache lookups monitor element

The number of times that an application looked for a section or package in the package cache. At a database level, it indicates the overall number of references since the database was started, or monitor data was reset. This counter includes the cases where the section is already loaded in the cache and when the section has to be loaded into the cache. In a concentrator environment where agents are being associated with different applications, additional package cache lookups may be required as a result of a new agent not having the required section or package available in local storage.

Table 1162. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1163. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1164. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Database     | event_db  | Always collected     |
| Connection   | event_conn  | Always collected     |
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## Usage

To calculate the package cache hit ratio use the following formula:

$$1 - (\text{Package Cache Inserts} / \text{Package Cache Lookups})$$

The package cache hit ratio tells you whether or not the package cache is being used effectively. If the hit ratio is high (more than 0.8), the cache is performing well. A smaller hit ratio may indicate that the package cache should be increased.

You will need to experiment with the size of the package cache to find the optimal number for the **pkcachesz** configuration parameter. For example, you might be able to use a smaller package cache size if there is no increase in the **pkg\_cache\_inserts** element when you decrease the size of the cache. Decreasing the package cache size frees up system resources for other work. It is also possible that you could improve overall system performance by increasing the size of the package cache if by doing so, you decrease the number of **pkg\_cache\_inserts**. This experimentation is best done under full workload conditions.

You can use this element with **ddl\_sql\_stmts** to determine whether or not the execution of DDL statements is impacting the performance of the package cache. Sections for dynamic SQL statements can become invalid when DDL statements are executed. Invalid sections are implicitly prepared by the system when next used. The execution of a DDL statement could invalidate a number of sections and the resulting additional processing time required when preparing those sections could significantly impact performance. In this case, the package cache hit ratio reflects the implicit recompilation of invalid sections. It does not reflect the insertion of new sections into the cache, so increasing the size of the package cache will not improve overall performance. You might find it less confusing to tune the cache for an application on its own before working in the full environment.

It is necessary to determine the role that DDL statements are playing in the value of the package cache hit ratio before deciding on what action to take. If DDL statements rarely occur, then cache performance may be improved by increasing its

size. If DDL statements are frequent, then improvements may require that you limit the use of DDL statements (possibly to specific time periods).

The **static\_sql\_stmts** and **dynamic\_sql\_stmts** counts can be used to help provide information about the quantity and type of sections being cached.

**Note:** You may want to use this information at the database level to calculate the average package cache hit ratio all each applications. You should look at this information at an application level to find out the exact package cache hit ratio for a given application. It may not be worthwhile to increase the size of the package cache in order to satisfy the cache requirements of an application that only executes infrequently.

---

## pkg\_cache\_num\_overflows - Package Cache Overflows

The number of times that the package cache overflowed the bounds of its allocated memory.

*Table 1165. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1166. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

### Usage

Use this element with the **pkg\_cache\_size\_top** monitor element to determine whether the size of the package cache needs to be increased to avoid overflowing.

---

## pkg\_cache\_size\_top - Package cache high watermark

The largest size reached by the package cache.

**Note:** The **pkg\_cache\_size\_top** monitor element is deprecated starting with DB2 Version 9.5. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1167. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 1168. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

## Usage

If the package cache overflowed, then this element contains the largest size reached by the package cache during the overflow.

Check the **pkg\_cache\_num\_overflows** monitor element to determine if such a condition occurred.

You can determine the minimum size of the package cache required by your workload by:

maximum package cache size / 4096

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the package cache to avoid overflow.

---

## pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element

The number of group buffer pool (GBP) independent data pages found in a local buffer pool by asynchronous EDUs.

*Table 1169. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

*Table 1170. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

---

## pool\_async\_data\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid data pages monitor element

The number of times a data page was attempted to be read from the group buffer pool by a prefetcher because the page was invalid in the local buffer pool.

*Table 1171. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads} \right) / \text{pool\_async\_data\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

### **pool\_async\_data\_gbp\_l\_reads - Asynchronous group buffer pool data logical reads monitor element**

The number of times a Group Buffer Pool (GBP) dependent data page was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 1172. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads} \right) / \text{pool\_async\_data\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

### **pool\_async\_data\_gbp\_p\_reads - Asynchronous group buffer pool data physical reads monitor element**

The number of times a Group Buffer Pool (GBP) dependent data page was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

*Table 1173. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}}{\text{pool\_async\_data\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_data\_lbp\_pages\_found - Asynchronous local buffer pool data pages found monitor element

The number of times a data page was present in the local buffer pool when a prefetcher attempted to access it.

*Table 1174. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}}{\text{pool\_async\_data\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_data\_read\_reqs - Buffer pool asynchronous read requests monitor element

The number of asynchronous read requests by the prefetcher to the operating system. These requests are typically large block I/Os of multiple pages.

*Table 1175. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE                     |

*Table 1176. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1177. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

## Usage

To calculate the average number of data pages in each read request, use the following formula:

$$\text{pool\_async\_data\_reads} / \text{pool\_async\_data\_read\_reqs}$$

This average can help you determine the average read I/O size used by the prefetcher. This data can also be helpful in understanding the large block I/O requirements of the measured workload.

The maximum size of a prefetcher read I/O is the value specified on the EXTENTSIZE option of the CREATE TABLESPACE statement for the table space involved, but it can be smaller under some circumstances:

- when some pages of the extent are already in the buffer pool
- when exceeding operating system capabilities
- when the EXTENTSIZE option value is very large, such that doing a large I/O would be detrimental to overall performance

---

## pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element

Indicates the number of data pages read in from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces.

Table 1178. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

Table 1179. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1180. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

## Usage

You can use this element with **pool\_data\_p\_reads** to calculate the number of physical reads that were performed synchronously (that is, physical data page reads that were performed by database manager agents). Use the following formula:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads}) - (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads})) / (\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads})$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num\_ioservers** configuration parameter.

Asynchronous reads are performed by database manager prefetchers.

---

## pool\_async\_data\_writes - Buffer pool asynchronous data writes monitor element

The number of times a buffer pool data page was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher may have written dirty pages to disk to make space for the pages being prefetched.

*Table 1181. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

*Table 1182. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1183. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

**Usage** You can use this element with the **pool\_data\_writes** monitor element to calculate the number of physical write requests that were performed synchronously (that is, physical data page writes that were performed by database manager agents). Use the following formula:

$$\text{pool\_data\_writes} - \text{pool\_async\_data\_writes}$$

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the `num_iocleaners` configuration parameter.

---

## pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element

The number of group buffer pool (GBP) independent index pages found in a local buffer pool by asynchronous EDUs.

*Table 1184. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

*Table 1185. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

---

## pool\_async\_index\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid index pages monitor element

The number of times an index page was attempted to be read from the group buffer pool by a prefetcher because the page was invalid in the local buffer pool.

*Table 1186. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}}{\text{pool\_async\_index\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_gbp\_l\_reads - Asynchronous group buffer pool index logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 1187. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}}{\text{pool\_async\_index\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_gbp\_p\_reads - Asynchronous group buffer pool index physical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

*Table 1188. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}}{\text{pool\_async\_index\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_lbp\_pages\_found - Asynchronous local buffer pool index pages found monitor element

The number of times an index page was present in the local buffer pool when a prefetcher attempted to access it.

*Table 1189. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}}{\text{pool\_async\_index\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_read\_reqs - Buffer pool asynchronous index read requests monitor element

The number of asynchronous read requests for index pages.

*Table 1190. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE                     |

*Table 1191. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1192. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

**Usage** To calculate the number of index pages read per asynchronous request, use the following formula:

$$\text{pool\_async\_index\_reads} / \text{pool\_async\_index\_read\_reqs}$$

This average can help you determine the amount of asynchronous I/O done for index pages in each interaction with the prefetcher.

---

## pool\_async\_index\_reads - Buffer pool asynchronous index reads monitor element

Indicates the number of index pages read in from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces.

*Table 1193. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

*Table 1194. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1195. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

### Usage

You can use this element with the **pool\_index\_p\_reads** monitor element to calculate the number of physical reads that were performed synchronously (that is, physical index page reads that were performed by database manager agents). Use the following formula:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads}) - (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads})) / (\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads})$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num\_ioservers** configuration parameter.

Asynchronous reads are performed by database manager prefetchers.

---

## pool\_async\_index\_writes - Buffer pool asynchronous index writes monitor element

The number of times a buffer pool index page was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher may have written dirty pages to disk to make space for the pages being prefetched.

Table 1196. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE                     |

Table 1197. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1198. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

### Usage

You can use this element with the **pool\_index\_writes** monitor element to calculate the number of physical index write requests that were performed synchronously (that is, physical index page writes that were performed by database manager agents). Use the following formula:

$$\text{pool\_index\_writes} - \text{pool\_async\_index\_writes}$$

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the **num\_iocleaners** configuration parameter.

---

## pool\_async\_read\_time - Buffer Pool Asynchronous Read Time

Indicates the total amount of time spent reading in data and index pages from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces. This value is given in milliseconds.

### Element identifier

pool\_async\_read\_time

### Element type

counter

Table 1199. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

Table 1200. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1201. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

**Usage** You can use this element to calculate the elapsed time for synchronous reading, using the following formula:

$$\text{pool\_read\_time} - \text{pool\_async\_read\_time}$$

You can also use this element to calculate the average asynchronous read time using the following formula:

$$\text{pool\_async\_read\_time} / \text{pool\_async\_data\_reads}$$

These calculations can be used to understand the I/O work being performed.

## pool\_async\_write\_time - Buffer pool asynchronous write time monitor element

The total elapsed time spent writing data or index pages from the buffer pool to disk by database manager page cleaners. This value is reported in milliseconds.

Table 1202. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

Table 1203. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |

Table 1203. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1204. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

## Usage

To calculate the elapsed time spent writing pages synchronously, use the following formula:

$$\text{pool\_write\_time} - \text{pool\_async\_write\_time}$$

You can also use this element to calculate the average asynchronous write time using the following formula:

$$\frac{\text{pool\_async\_write\_time}}{(\text{pool\_async\_data\_writes} + \text{pool\_async\_index\_writes})}$$

These calculations can be used to understand the I/O work being performed.

---

## **pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element**

The number of group buffer pool (GBP) independent XML storage object (XDA) pages found in a local buffer pool by asynchronous EDUs.

Table 1205. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

Table 1206. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

---

## pool\_async\_xda\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid XDA data pages monitor element

The number of times a request for a data page for an XML storage object (XDA) was made from the group buffer pool by a prefetcher due to the page being marked invalid in the local buffer pool.

*Table 1207. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_gbp\_l\_reads - Group buffer pool XDA data asynchronous logical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 1208. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_gbp\_p\_reads - Group buffer pool XDA data asynchronous physical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

Table 1209. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads} \right) / \text{pool\_async\_xda\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_lbp\_pages\_found - Asynchronous local buffer pool XDA data pages found monitor element

The number of times a data page for an XML storage object (XDA) was requested by a prefetcher from and found in the local buffer pool.

Table 1210. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads} \right) / \text{pool\_async\_xda\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_read\_reqs - Buffer pool asynchronous XDA read requests monitor element

The number of asynchronous read requests for XML storage object (XDA) data.

*Table 1211. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE                     |

*Table 1212. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1213. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

**Usage** To calculate the average number of XML storage object data pages read per asynchronous request, use the following formula:

$$\text{pool\_async\_xda\_reads} / \text{pool\_async\_xda\_read\_reqs}$$

This average can help you determine the amount of asynchronous I/O done in each interaction with the prefetcher.

---

## pool\_async\_xda\_reads - Buffer pool asynchronous XDA data reads monitor element

Indicates the number of XML storage object (XDA) data pages read in from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces.

*Table 1214. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE                     |

Table 1215. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1216. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

## Usage

Use the **pool\_async\_xda\_reads** and **pool\_xda\_p\_reads** monitor elements to calculate the number of physical reads that were performed synchronously on XML storage object data pages (that is, physical data page reads that were performed by database manager agents on XML data). Use the following formula:

$$\text{pool\_xda\_p\_reads} - \text{pool\_async\_xda\_reads}$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num\_ioservers** configuration parameter.

Asynchronous reads are performed by database manager prefetchers.

---

## pool\_async\_xda\_writes - Buffer pool asynchronous XDA data writes monitor element

The number of times a buffer pool data page for an XML storage object (XDA) was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher may have written dirty pages to disk to make space for the pages being prefetched.

Table 1217. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE                     |

Table 1218. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1219. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |

**Usage** You can use this element with the **pool\_xda\_writes** monitor element to calculate the number of physical write requests that were performed synchronously on XML storage object data pages (that is, physical data page writes that were performed by database manager agents on XML data). Use the following formula:

$$\text{pool\_xda\_writes} - \text{pool\_async\_xda\_writes}$$

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the **num\_iocleaners** configuration parameter.

---

## pool\_config\_size - Configured Size of Memory Pool

The internally configured size of a memory pool in DB2 database system. The value is given in bytes.

Table 1220. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 1221. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbmemuse        | Always collected |
| Connection | event_connmemuse      | Always collected |

**Usage** To track system memory usage, use this value in conjunction with **pool\_cur\_size**, **pool\_id**, and **pool\_watermark**.

To see if a memory pool is nearly full, compare **pool\_config\_size** to **pool\_cur\_size**. For example, assume that the utility heap is too small. You can diagnose this specific problem by taking snapshots at regular intervals, and looking in the utility heap section of the snapshot output. If required, the **pool\_cur\_size** might be allowed to exceed the **pool\_config\_size** to prevent an out of memory failure. If this occurs very infrequently, no further action is likely required. However if **pool\_cur\_size** is consistently close to or larger than **pool\_config\_size**, you might consider increasing the size of the utility heap.

---

## pool\_cur\_size - Current Size of Memory Pool

The current size of a memory pool. The value is given in bytes.

Table 1222. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 1223. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbmemuse        | Always collected |
| Connection | event_connmemuse      | Always collected |

**Usage** To track system memory usage, use this value in conjunction with *pool\_config\_size*, *pool\_id*, and *pool\_watermark*.

To see if a memory pool is nearly full, compare *pool\_config\_size* to *pool\_cur\_size*. For example, assume that the utility heap is too small. You can diagnose this specific problem by taking snapshots at regular intervals, and looking in the utility heap section of the snapshot output. If the value of *pool\_cur\_size* is consistently close to *pool\_config\_size*, you may want to consider increasing the size of the utility heap.

---

## pool\_data\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent data pages found in local buffer pool monitor element

The number of group buffer pool (GBP) independent data pages found in a local buffer pool (LBP) by an agent.

Table 1224. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1224. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache<br>entries (reported in DETAILS XML<br>document)         | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1225. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

## pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element

The number of times a data page was invalid in the local buffer pool and was read from the group buffer pool instead. Outside of a DB2 pureScale environment, this value is null.

Table 1226. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics                             | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics   | DATA OBJECT METRICS BASE   |

Table 1226. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1227. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -  | Always collected      |
| Package cache | -  | Always collected      |
| Locking       | -  | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was attempted to be read from the group buffer pool because the page was either invalid or not present in the Local Buffer Pool (LBP).

*Table 1228. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1228. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1229. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was read into the local buffer pool from disk because it was not found in the GBP.

Table 1230. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |

Table 1230. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element     | Collection Level |
|---|---------------------|------------------|
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS     | BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function -<br>- Get unit of work metrics   | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS     | BASE             |

Table 1231. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -  | Always collected      |
| Package cache | -  | Always collected      |
| Locking       | -  | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element

The number of times a data page was present in the local buffer pool.

*Table 1232. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1233. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_data\_l\_reads - Buffer pool data logical reads monitor element

The number of data pages which have been requested from the buffer pool (logical) for regular and large table spaces.

Table 1234. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1234. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1235. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1236. Event Monitoring Information

| Event Type   | Logical Data Grouping                                    | Monitor Switch        |
|--------------|--|-----------------------|
| Activities   | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Database     | event_db   | Always collected      |

Table 1236. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch         |
|---------------|--|------------------------|
| Tablespaces   | event_tablespace                           | Always collected       |
| Connection    | event_conn                                 | Always collected       |
| Statement     | event_stmt                                 | Always collected       |
| Activities    | event_activity                             | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE  |

## Usage

This count includes accesses to data that is:

- Already in the buffer pool when the database manager needs to process the page.
- Read into the buffer pool before the database manager can process the page.

You can use the **pool\_data\_l\_reads** and **pool\_data\_p\_reads** monitor elements to calculate the data page hit ratios. For example, the following formula returns the data page hit ratios in a DB2 environment without the IBM DB2 pureScale Feature:

$$\frac{(\text{pool\_data\_lbp\_pages\_found} - \text{pool\_async\_data\_lbp\_pages\_found} - \text{pool\_temp\_data\_l\_reads})}{\text{pool\_data\_l\_reads}} \times 100$$

For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

Increasing buffer pool size will generally improve the hit ratio, but you will reach a point of diminishing return. Ideally, if you could allocate a buffer pool large enough to store your entire database, then once the system is up and running you would get a hit ratio of 100%. However, this is unrealistic in most cases. The significance of the hit ratio really depends on the size of your data, and the way it is accessed. A very large database where data is accessed evenly would have a poor hit ratio. There is little you can do with very large tables.

To improve hit ratios for smaller, frequently accessed tables and indexes, assign them to individual buffer pools.

---

## pool\_data\_p\_reads - Buffer pool data physical reads monitor element

Indicates the number of data pages read in from the table space containers (physical) for regular and large table spaces.

Table 1237. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |

Table 1237. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1238. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1239. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 1239. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch         |
|---------------|--|------------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE   |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE   |
| Database      | event_db   | Always collected       |
| Tablespaces   | event_tablespace                                     | Always collected       |
| Connection    | event_conn   | Always collected       |
| Statement     | event_stmt   | Always collected       |
| Activities    | event_activity                                       | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE  |

## Usage

Use this element with the **pool\_data\_l\_reads** and **pool\_async\_data\_reads** monitor elements to calculate the number of physical reads that were performed synchronously (that is, physical data page reads that were performed by database manager agents). Use the following formula:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads}) - (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads})) / (\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads})$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This information can be helpful when you are tuning the **num\_ioservers** configuration parameter.

## pool\_data\_writes - Buffer pool data writes monitor element

The number of times a buffer pool data page was physically written to disk.

Table 1240. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1240. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1241. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1242. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database     | event_db  | Always collected      |

Table 1242. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Tablespaces   | event_tablespace                           | Always collected      |
| Connection    | event_conn                                 | Always collected      |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

If a buffer pool data page is written to disk for a high percentage of the value of the **pool\_data\_p\_reads** monitor element, you may be able to improve performance by increasing the number of buffer pool pages available for the database.

A buffer pool data page is written to disk for the following reasons:

- To free a page in the buffer pool so another page can be read
- To flush the buffer pool

The system does not always write a page to make room for a new one. If the page has not been updated, it can simply be replaced. This replacement is not counted for this element.

The data page can be written by an asynchronous page-cleaner agent before the buffer pool space is required, as reported by the **pool\_async\_data\_writes** monitor element. These asynchronous page writes are included in the value of this element in addition to synchronous page writes.

When calculating this percentage, disregard the number of physical reads required to initially fill the buffer pool. To determine the number of pages written:

1. Run your application (to load the buffer).
2. Note the value of this element.
3. Run your application again.
4. Subtract the value recorded in step 2 from the new value of this element.

In order to prevent the buffer pool from being deallocated between the runnings of your application, you should do one of the following:

- Activate the database with the **ACTIVATE DATABASE** command.
- Have an idle application connected to the database.

If all applications are updating the database, increasing the size of the buffer pool may not have much impact on performance since most of the buffer pool pages contain updated data, which must be written to disk. However, if the updated pages can be used by other units of work before being written out, the buffer pool can save a write and a read, which will improve your performance.

---

## pool\_drty\_pg\_steal\_clns - Buffer pool victim page cleaners triggered monitor element

The number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

Table 1243. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |

Table 1244. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1245. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

### Usage

Using the following formula, you may calculate what percentage of all cleaner invocations are represented by this element:

$$\frac{\text{pool\_drty\_pg\_steal\_clns}}{(\text{pool\_drty\_pg\_steal\_clns} + \text{pool\_drty\_pg\_thrsh\_clns} + \text{pool\_l\_sn\_gap\_clns})}$$

If this ratio is low, it may indicate that you have defined too many page cleaners. If your **chnpggs\_thresh** configuration parameter is set too low, you may be writing out pages that you will dirty later. Aggressive cleaning defeats one purpose of the buffer pool, that is to defer writing to the last possible moment.

If this ratio is high, it may indicate that you have not defined enough page cleaners. Not having enough page cleaners increases recovery time after failures.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is OFF:

- The **pool\_drty\_pg\_steal\_clns** monitor element is inserted into the monitor stream.
- The **pool\_drty\_pg\_steal\_clns** monitor element counts the number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is ON:

- The **pool\_drty\_pg\_steal\_clns** monitor element inserts 0 into the monitor stream.
- There is no explicit triggering of the page cleaners when a synchronous write is needed during victim buffer replacement. To determine whether or not the right number of page cleaners is configured for the database or for specific buffer pools, refer to the **pool\_no\_victim\_buffer** monitor element.

**Note:** Although dirty pages are written out to disk, the pages are not removed from the buffer pool right away, unless the space is needed to read in new pages.

## pool\_drty\_pg\_thrsh\_clns - Buffer pool threshold cleaners triggered monitor element

The number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

Table 1246. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |

Table 1247. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1248. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** The threshold is set by the **chnpggs\_thrsh** configuration parameter. It is a percentage applied to the buffer pool size. When the number of dirty pages in the pool exceeds this value, the cleaners are triggered.

If the **chnpggs\_thrsh** configuration parameter value is set too low, pages might be written out too early, requiring them to be read back in. If it is set too high, then too many pages may accumulate, requiring users to write out pages synchronously.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is OFF:

- The **pool\_drty\_pg\_thrsh\_clns** monitor element is inserted into the monitor stream.
- The **pool\_drty\_pg\_thrsh\_clns** monitor element counts the number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is ON:

- The **pool\_drty\_pg\_thrsh\_clns** monitor element inserts 0 into the monitor stream.
- Page cleaners are always active, attempting to ensure there are sufficient free buffers for victims available instead of waiting to be triggered by the criterion value.

## pool\_failed\_async\_data\_reqs - Failed data prefetch requests monitor element

The number of times an attempt to queue a data prefetch request was made but failed. One possible reason is the prefetch queue is full.

Table 1249. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1250. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async\_...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) × 100

```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter **num\_ioservers**. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements **pool\_queued\_async\_...\_pages** and **pool\_queued\_async\_...\_reqs** to determine the average prefetch request size.

## pool\_failed\_async\_index\_reqs - Failed index prefetch requests monitor element

The number of times an attempt to queue an index prefetch request was made but failed. One possible reason is the prefetch queue is full and a request could not be obtained from the free list.

*Table 1251. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |

Table 1251. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1252. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +

```

```

        POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
        POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
+
    (
        POOL_QUEUED_ASYNC_DATA_REQS +
        POOL_QUEUED_ASYNC_INDEX_REQS +
        POOL_QUEUED_ASYNC_XDA_REQS +
        POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
        POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
        POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
) × 100

```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter **num\_ioservers**. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements **pool\_queued\_async...\_pages** and **pool\_queued\_async...\_reqs** to determine the average prefetch request size.

---

## pool\_failed\_async\_other\_reqs - Failed non-prefetch requests monitor element

The number of times an attempt to queue a non-prefetch request was made but failed. This element is for non-prefetch work done by prefetchers. One possible reason for the failed request is that the prefetch queue is full.

*Table 1253. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 1253. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1254. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element reports the number of requests for work that is not related to the prefetching dictated by an access plan that could not be added to the prefetch queue. Utilities like the backup utility use the prefetcher mechanism to perform their tasks, but in a way that is different from the way an access plan for an SQL statement does. A request might fail to be added to a prefetch queue because the queue is full.

## pool\_failed\_async\_temp\_data\_reqs - Failed data prefetch requests for temporary table spaces monitor element

The number of times an attempt to queue a data prefetch request for temporary table spaces was made but failed. One possible reason is the prefetch queue is full and a request could not be obtained from the free list.

Table 1255. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1256. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async\_...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) × 100

```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter **num\_ioservers**. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements **pool\_queued\_async\_...\_pages** and **pool\_queued\_async\_...\_reqs** to determine the average prefetch request size.

---

## pool\_failed\_async\_temp\_index\_reqs - Failed index prefetch requests for temporary table spaces monitor element

The number of times an attempt to queue an index prefetch request for temporary table spaces was made but failed. One possible reason is the prefetch queue is full and a request could not be obtained from the free list.

*Table 1257. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |

Table 1257. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1258. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +

```

```

    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
+
  (
    POOL_QUEUED_ASYNC_DATA_REQS +
    POOL_QUEUED_ASYNC_INDEX_REQS +
    POOL_QUEUED_ASYNC_XDA_REQS +
    POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
    POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
    POOL_QUEUED_ASYNC_TEMP_XDA_REQS
  )
) × 100

```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter `num_ioservers`. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements `pool_queued_async..._pages` and `pool_queued_async..._reqs` to determine the average prefetch request size.

---

## pool\_failed\_async\_temp\_xda\_reqs - Failed XDA prefetch requests for temporary table spaces monitor element

The number of times an attempt to queue a XML storage object (XDA) data prefetch request for temporary table spaces was made but failed. One possible reason is the prefetch queue is full and a request could not be obtained from the free list.

*Table 1259. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 1259. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1260. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async\_...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

$$1 - \frac{\text{POOL\_FAILED\_ASYNC\_DATA\_REQS}}{\text{POOL\_FAILED\_ASYNC\_DATA\_REQS} + \dots}$$

```

POOL_FAILED_ASYNC_INDEX_REQS +
POOL_FAILED_ASYNC_XDA_REQS +
POOL_FAILED_ASYNC_TEMP_DATA_REQS +
POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
POOL_FAILED_ASYNC_TEMP_XDA_REQS
)
÷
(
(
POOL_FAILED_ASYNC_DATA_REQS +
POOL_FAILED_ASYNC_INDEX_REQS +
POOL_FAILED_ASYNC_XDA_REQS +
POOL_FAILED_ASYNC_TEMP_DATA_REQS +
POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
POOL_FAILED_ASYNC_TEMP_XDA_REQS
)
+
(
POOL_QUEUED_ASYNC_DATA_REQS +
POOL_QUEUED_ASYNC_INDEX_REQS +
POOL_QUEUED_ASYNC_XDA_REQS +
POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
POOL_QUEUED_ASYNC_TEMP_XDA_REQS
)
) × 100

```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter **num\_ioservers**. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements **pool\_queued\_async\_...\_pages** and **pool\_queued\_async\_...\_reqs** to determine the average prefetch request size.

---

## pool\_failed\_async\_xda\_reqs - Failed XDA prefetch requests monitor element

The number of times an attempt to queue a XML storage object (XDA) data prefetch request was made but failed. One possible reason is the prefetch queue is full and a request could not be obtained from the free list.

*Table 1261. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 1261. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element     | Collection Level |
|---|---------------------|------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS    | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS     | BASE             |

Table 1262. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_failed\_async...\_reqs** elements tells you how many prefetch requests could not be added to a prefetch queue. Requests can fail to be added to the prefetch queue if the prefetch queue is too small, or if the prefetcher is running too slowly. When requests cannot be added to a prefetch queue, a database agent typically performs disk IO synchronously, which is less efficient than prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```
1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) × 100
```

This formula calculates the ratio of successful prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. Requests can fail to be added to the prefetch queue if there are a large number of requests being created, or if the prefetcher is running too slowly due to poor configuration or poor tuning. If the percentage of successful requests is low, this can indicate a bottleneck in the prefetching mechanism. You might need to configure more prefetchers by modifying the value for the configuration parameter **num\_ioservers**. The condition of prefetch queues being full can also be caused by agents submitting too many small requests; you can use the related monitor elements **pool\_queued\_async...\_pages** and **pool\_queued\_async...\_reqs** to determine the average prefetch request size.

---

## pool\_id - Memory Pool Identifier

The type of memory pool.

*Table 1263. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |

Table 1263. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | memory_pool           | Basic          |
| Application    | memory_pool           | Basic          |

Table 1264. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_dbmemuse        | -              |
| Connection | event_connmemuse      | -              |

## Usage

To track system memory usage, use this value in conjunction with **pool\_config\_size**, **pool\_cur\_size**, and **pool\_watermark**.

Use **pool\_id** to identify the memory pools discussed in the system monitor output. The various memory pool identifiers can be found in `sqlmon.h`. Under normal operating conditions, one or more of each of the following pools can be expected.

| API Constant            | Description                 |
|-------------------------|-----------------------------|
| SQLM_HEAP_APPLICATION   | Application Heap            |
| SQLM_HEAP_DATABASE      | Database Heap               |
| SQLM_HEAP_LOCK_MGR      | Lock Manager Heap           |
| SQLM_HEAP_UTILITY       | Backup/Restore/Utility Heap |
| SQLM_HEAP_STATISTICS    | Statistics Heap             |
| SQLM_HEAP_PACKAGE_CACHE | Package Cache Heap          |
| SQLM_HEAP_CAT_CACHE     | Catalog Cache Heap          |
| SQLM_HEAP_MONITOR       | Database Monitor Heap       |
| SQLM_HEAP_STATEMENT     | Statement Heap              |
| SQLM_HEAP_FCMBP         | FCMBP Heap                  |
| SQLM_HEAP_IMPORT_POOL   | Import Pool                 |
| SQLM_HEAP_OTHER         | Other Memory                |
| SQLM_HEAP_BP            | Buffer Pool Heap            |
| SQLM_HEAP_APPL_SHARED   | Applications Shared Heap    |
| SQLM_HEAP_SHARED_SORT   | Sort Shared Heap            |

## pool\_index\_gbp\_indep\_pages \_found\_in\_lbp - Group buffer pool independent index pages found in local buffer pool monitor element

The number of group buffer pool (GBP) independent index pages found in a local buffer pool (LBP) by an agent.

*Table 1265. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries (reported in DETAILS XML document)         | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1266. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

## pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element

The number of times an index page was attempted to be read from the group bufferpool because the page was invalid in the local buffer pool.

Table 1267. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1268. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was attempted to be read from the group buffer pool because the page was either invalid or not present in the local buffer pool.

Table 1269. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1269. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS     | BASE             |

Table 1270. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help

you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements

The number of times a Group Buffer Pool (GBP) dependent index page was read into the local buffer pool from disk because it was not found in the GBP.

Table 1271. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1272. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |

Table 1272. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activitymetrics                                | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | -  | Always collected      |
| Package cache | -  | Always collected      |
| Locking       | -  | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element

The number of times an index page was present in the local buffer pool.

Table 1273. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1273. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1274. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(\text{POOL\_INDEX\_LBP\_PAGES\_FOUND} - \text{POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND}) / \text{POOL\_INDEX\_L\_READS}$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(\text{POOL\_INDEX\_GBP\_L\_READS} - \text{POOL\_INDEX\_GBP\_P\_READS}) / \text{POOL\_INDEX\_GBP\_L\_READS}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_index\_l\_reads - Buffer pool index logical reads monitor element

Indicates the number of index pages which have been requested from the buffer pool (logical) for regular and large table spaces.

*Table 1275. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1276. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

Table 1276. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1277. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

This count includes accesses to index pages that are:

- Already in the buffer pool when the database manager needs to process the page
- Read into the buffer pool before the database manager can process the page.

In conjunction with **pool\_index\_p\_reads**, and **pool\_async\_index\_reads** you can use **pool\_index\_l\_reads** to calculate the index page hit ratio for the buffer pool. For example, the following formula returns the index page hit ratios in a DB2 environment without the IBM DB2 pureScale Feature

$$\frac{((\text{pool\_index\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_temp\_index\_l\_reads})}{\text{pool\_index\_l\_reads}} \times 100$$

For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

If the hit ratio is low, increasing the number of buffer pool pages may improve performance.

## pool\_index\_p\_reads - Buffer pool index physical reads monitor element

Indicates the number of index pages read in from the table space containers (physical) for regular and large table spaces.

*Table 1278. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1279. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |

Table 1279. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1280. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

In conjunction with the **pool\_index\_l\_reads**, and **pool\_async\_index\_reads** you can use **pool\_index\_p\_reads** to calculate the index page hit ratio for the buffer pool. For example, the following formula returns the index page hit ratios in a DB2 environment without the IBM DB2 pureScale Feature

$$\frac{((\text{pool\_index\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_temp\_index\_l\_reads})}{\text{pool\_index\_l\_reads}} \times 100$$

For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

## pool\_index\_writes - Buffer pool index writes monitor element

Indicates the number of times a buffer pool index page was physically written to disk.

*Table 1281. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1282. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

Table 1282. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1283. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Connection    | event_conn  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Like a data page, a buffer pool index page is written to disk for the following reasons:

- To free a page in the buffer pool so another page can be read
- To flush the buffer pool

The system does not always write a page to make room for a new one. If the page has not been updated, it can simply be replaced. This replacement is not counted for this element.

The index page can be written by an asynchronous page-cleaner agent before the buffer pool space is required. These asynchronous index page writes are included in the value of this element in addition to synchronous index page writes (see the **pool\_async\_index\_writes** monitor element).

If a buffer pool index page is written to disk for a high percentage of the value of the **pool\_index\_p\_reads** monitor element, you may be able to improve performance by increasing the number of buffer pool pages available for the database.

When calculating this percentage, disregard the number of physical reads required to initially fill the buffer pool. To determine the number of pages written:

1. Run your application (to load the buffer).
2. Note the value of this element.
3. Run your application again.
4. Subtract the value recorded in step 2 from the new value of this element.

In order to prevent the buffer pool from being deallocated between the runnings of your application, you should do one of the following:

- Activate the database with the **ACTIVATE DATABASE** command.
- Have an idle application connected to the database.

If all applications are updating the database, increasing the size of the buffer pool may not have much impact on performance, since most of the pages contain updated data which must be written to disk.

---

## pool\_lsn\_gap\_clns - Buffer pool log space cleaners triggered monitor element

The number of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.

*Table 1284. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |

*Table 1285. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1286. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

### Usage

This element can be used to help evaluate whether you have enough space for logging, and whether you need more log files or larger log files.

The page cleaning criterion is determined by the setting for the **softmax** configuration parameter. Page cleaners are triggered if the oldest page in the buffer pool contains an update described by a log record that is older than the current log position by the criterion value.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is OFF:

- The **pool\_lsn\_gap\_clns** monitor element is inserted into the monitor stream.
- Page cleaners are triggered if the oldest page in the buffer pool contains an update described by a log record that is older than the current log position by the criterion value.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is ON:

- The **pool\_lsn\_gap\_clns** monitor element inserts 0 into the monitor stream.
- Page cleaners write pages proactively instead of waiting to be triggered by the criterion value.

---

## pool\_no\_victim\_buffer - Buffer pool no victim buffers monitor element

Number of times an agent did not have a preselected victim buffer available.

*Table 1287. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |

*Table 1288. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Tablespace     | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 1289. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Tablespace | event_tablespace      | Always collected |

**Usage** This element can be used to help evaluate whether you have enough page cleaners for a given buffer pool when using proactive page cleaning.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is ON, the pool\_no\_victim\_buffer element counts the number of times that an agent did not find a preselected victim buffer available for immediate use, and was forced to search the buffer pool for a suitable victim buffer.

If the value of pool\_no\_victim\_buffer element is high relative to the number of logical reads in the buffer pool, then the DB2 database system is having difficulty ensuring that sufficient numbers of good victims are available for use. Increasing the number of page cleaners will increase the ability of DB2 to provide preselected victim buffers.

When the DB2\_USE\_ALTERNATE\_PAGE\_CLEANING registry variable is OFF, the pool\_no\_victim\_buffer element has no predictive value, and can be safely ignored. In this configuration, the DB2 database system does not attempt to ensure that agents have preselected victim buffers available to them, so most accesses to the buffer pool will require that the agent search the buffer pool to find a victim buffer.

---

## pool\_queued\_async\_data\_pages - Data pages prefetch requests monitor element

The number of data pages successfully requested for prefetching.

*Table 1290. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1290. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE            |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE             |

Table 1291. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |

Table 1291. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |

## Usage

This monitor element, along with the other **pool\_queued\_async...pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES})}{(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS})}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...reqs** monitor element

## pool\_queued\_async\_data\_reqs - Data prefetch requests monitor element

The number of data prefetch requests successfully added to the prefetch queue.

Table 1292. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |

Table 1292. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_CONNECTION table function -<br>Get connection metrics  | REQUEST METRICS     | BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                       | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                                      | ACTIVITY METRICS    | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries (reported in DETAILS<br>XML document) | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics  | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document)        | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics   | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)                | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics  | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                           | REQUEST METRICS     | BASE             |

Table 1293. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>details_xml document)<br>event_activitiymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the<br>metrics.xml document)<br>pkgcache_metrics         | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the<br>metrics.xml document)<br>uow_metrics                   | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the<br>details_xml document)                        | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)                        | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\_\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```
1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) * 100
```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more prefetchers by modifying the **num\_ioservers** configuration parameter.

---

## pool\_queued\_async\_index\_pages - Index pages prefetch requests monitor element

The number of index pages successfully requested for prefetching.

*Table 1294. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |

Table 1294. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_CONNECTION table function -<br>Get connection metrics  | REQUEST METRICS     | BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                       | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                                      | ACTIVITY METRICS    | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries (reported in DETAILS<br>XML document) | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics  | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document)        | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics   | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)                | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics  | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                           | REQUEST METRICS     | BASE             |

Table 1295. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>details_xml document)<br>event_activitiymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the<br>metrics.xml document)<br>pkgcache_metrics         | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the<br>metrics.xml document)<br>uow_metrics                   | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the<br>details_xml document)                        | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)                        | REQUEST METRICS BASE  |

## Usage

This monitor element, along with the other **pool\_queued\_async...\_pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES})}{(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS})}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...\_reqs** monitor element

---

## pool\_queued\_async\_index\_reqs - Index prefetch requests monitor element

The number of index prefetch requests successfully added to the prefetch queue.

*Table 1296. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1296. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries (reported in DETAILS<br>XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document)        | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)                | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                           | REQUEST METRICS BASE             |

Table 1297. Event Monitoring Information

| Event Type    | Logical Data Grouping   | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in<br>details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the<br>metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the<br>metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the<br>details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
(
POOL_FAILED_ASYNC_DATA_REQS +
POOL_FAILED_ASYNC_INDEX_REQS +
POOL_FAILED_ASYNC_XDA_REQS +
POOL_FAILED_ASYNC_TEMP_DATA_REQS +
POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
POOL_FAILED_ASYNC_TEMP_XDA_REQS
)
÷
(
(
POOL_FAILED_ASYNC_DATA_REQS +
POOL_FAILED_ASYNC_INDEX_REQS +
POOL_FAILED_ASYNC_XDA_REQS +
POOL_FAILED_ASYNC_TEMP_DATA_REQS +
POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
POOL_FAILED_ASYNC_TEMP_XDA_REQS
)
+
(
POOL_QUEUED_ASYNC_DATA_REQS +
POOL_QUEUED_ASYNC_INDEX_REQS +
POOL_QUEUED_ASYNC_XDA_REQS +
POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
POOL_QUEUED_ASYNC_TEMP_XDA_REQS
)
) * 100

```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more prefetchers by modifying the **num\_ioservers** configuration parameter.

---

## pool\_queued\_async\_other\_reqs - Other requests handled by prefetchers monitor element

The number of requests for non-prefetch work successfully added to the prefetch queue. This is for other work done by prefetchers.

*Table 1298. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1298. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries (reported in DETAILS<br>XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document)        | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)                | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                           | REQUEST METRICS BASE             |

Table 1299. Event Monitoring Information

| Event Type    | Logical Data Grouping   | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in<br>details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the<br>metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the<br>metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the<br>details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element reports the number of requests added to the prefetch queue for IO work that is not related to the prefetching dictated by an access plan. Utilities like the backup utility use the prefetcher mechanism to perform their tasks, but in a way that is different from the way an access plan for an SQL statement does.

## pool\_queued\_async\_temp\_data\_pages - Data pages prefetch requests for temporary table spaces monitor element

The number of data pages for temporary table spaces successfully requested for prefetching.

*Table 1300. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1301. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element, along with the other **pool\_queued\_async...\_pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES}) \end{aligned}}{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS}) \end{aligned}}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...\_reqs** monitor element

## pool\_queued\_async\_temp\_data\_reqs - Data prefetch requests for temporary table spaces monitor element

The number of data prefetch requests for temporary table spaces successfully added to the prefetch queue.

Table 1302. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1303. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) * 100

```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more

prefetchers by modifying the `num_ioservers` configuration parameter.

## pool\_queued\_async\_temp\_index\_pages - Index pages prefetch requests for temporary table spaces monitor element

The number of index pages for temporary table spaces successfully requested for prefetching.

*Table 1304. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1305. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element, along with the other **pool\_queued\_async...\_pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES}) \end{aligned}}{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS}) \end{aligned}}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...\_reqs** monitor element

## pool\_queued\_async\_temp\_index\_reqs - Index prefetch requests for temporary table spaces monitor element

The number of index prefetch requests for temporary table spaces successfully added to the prefetch queue.

Table 1306. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1307. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) * 100

```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more

prefetchers by modifying the `num_ioservers` configuration parameter.

## pool\_queued\_async\_temp\_xda\_pages - XDA data pages prefetch requests for temporary table spaces monitor element

The number of XML storage object (XDA) data pages for temporary table spaces successfully requested for prefetching.

*Table 1308. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1309. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element, along with the other **pool\_queued\_async...\_pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES}) \end{aligned}}{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS}) \end{aligned}}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...\_reqs** monitor element

## pool\_queued\_async\_temp\_xda\_reqs - XDA data prefetch requests for temporary table spaces monitor element

The number of XML storage object (XDA) data prefetch requests for temporary table spaces successfully added to the prefetch queue.

Table 1310. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1311. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) * 100

```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more

prefetchers by modifying the `num_ioservers` configuration parameter.

## pool\_queued\_async\_xda\_pages - XDA pages prefetch requests monitor element

The number of XML storage object (XDA) data pages successfully requested for prefetching.

*Table 1312. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1313. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This monitor element, along with the other **pool\_queued\_async...\_pages** elements tells you how many pages of data were retrieved by prefetch requests. You can use this information to determine whether prefetch requests are being performed efficiently on your system. For example, you can calculate the average number of pages per prefetch request using a formula like the one that follows:

$$\frac{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_PAGES} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_PAGES}) \end{aligned}}{\begin{aligned} &(\text{POOL\_QUEUED\_ASYNC\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_XDA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_DATA\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_INDEX\_REQS} + \\ &\text{POOL\_QUEUED\_ASYNC\_TEMP\_XDA\_REQS}) \end{aligned}}$$

If the average number of pages per request is low, and there is a significant amount of prefetching on the system, then your system might be performing more IO operations than necessary. In general, request size is based on prefetch size, which should be at least as large as extent size. So a small average request size might indicate that prefetch size is set too low, and that increasing prefetch size to a multiple of extent size may improve performance. Also note that a small average request size may mean that the prefetch queues fill up too quickly, so it is worthwhile to also monitor the associated **pool\_failed\_async...\_reqs** monitor element

## pool\_queued\_async\_xda\_reqs - XDA prefetch requests monitor element

The number of XML storage object (XDA) data prefetch requests successfully added to the prefetch queue.

Table 1314. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1315. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## Usage

This element, along with the other **pool\_queued\_async\*\_reqs** elements tells you how many prefetch requests have been added to the prefetch queue. You can use this information to see how often the database manager does prefetching. You can use these elements in conjunction with other prefetcher monitor elements to determine how effectively prefetching is done on your system. For example, you can see what the percentage of requests were successfully added to the prefetch queue using a formula like the one that follows:

```

1 -
  (
    POOL_FAILED_ASYNC_DATA_REQS +
    POOL_FAILED_ASYNC_INDEX_REQS +
    POOL_FAILED_ASYNC_XDA_REQS +
    POOL_FAILED_ASYNC_TEMP_DATA_REQS +
    POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
    POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
÷
  (
    (
      POOL_FAILED_ASYNC_DATA_REQS +
      POOL_FAILED_ASYNC_INDEX_REQS +
      POOL_FAILED_ASYNC_XDA_REQS +
      POOL_FAILED_ASYNC_TEMP_DATA_REQS +
      POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
      POOL_FAILED_ASYNC_TEMP_XDA_REQS
    )
  +
    (
      POOL_QUEUED_ASYNC_DATA_REQS +
      POOL_QUEUED_ASYNC_INDEX_REQS +
      POOL_QUEUED_ASYNC_XDA_REQS +
      POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
      POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
      POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) * 100

```

This formula calculates the ratio of failed prefetch requests to the total number of requests made. A failed prefetch request is one that could not be added to the prefetch queue. If the percentage is low, you might need to configure more

prefetchers by modifying the `num_ioservers` configuration parameter.

## pool\_read\_time - Total buffer pool physical read time monitor element

Indicates the total amount of time spent reading in data and index pages from the table space containers (physical) for all types of table spaces. This value is given in milliseconds.

Table 1316. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 1316. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1317. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1318. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Connection    | event_conn  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

You can use this element with **pool\_data\_p\_reads** and **pool\_index\_p\_reads** monitor elements to calculate the average page-read time. This average is important since it may indicate the presence of an I/O wait, which in turn may indicate that you should be moving data to a different device.

At the database and table space levels, this element includes the value of the **pool\_async\_read\_time** monitor element.

---

## pool\_secondary\_id - Memory Pool Secondary Identifier

An additional identifier to help determine the memory pool for which monitor data is returned.

### Element identifier

pool\_secondary\_id

**Element type**  
Information

*Table 1319. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

*Table 1320. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbmemuse        | Always collected |
| Connection | event_connmemuse      | Always collected |

**Usage** Use together with `pool_id` to determine the memory pool for which monitor data is returned. Data for `pool_secondary_id` only appears when necessary. For example, it appears when the `pool_id` indicated is Buffer Pool Heap to determine which buffer pool the monitor data relates to.

When a database is created, it has a default buffer pool, called IBMDEFAULTBP, with a size determined by the platform. This buffer pool has a secondary id of "1". In addition to this buffer pool and any buffer pools that you create, a set of system buffer pools are created by default, each corresponding to a different page size. IDs for these buffer pools can appear in snapshots for `pool_secondary_id`:

- System 32k buffer pool
- System 16k buffer pool
- System 8k buffer pool
- System 4k buffer pool

---

## **pool\_sync\_data\_gbp\_reads - Synchronous group buffer pool data reads monitor element**

On a DB2 pureScale environment, the number of times a data page was expected to be in the bufferpool, but was instead retrieved from the group bufferpool. This value will be 0 for environments outside of a DB2 pureScale environment.

---

## **pool\_sync\_data\_reads - Synchronous buffer pool data reads monitor element**

The number of times a data page was expected to be in the bufferpool, but was instead read from disk.

---

## **pool\_sync\_index\_gbp\_reads - Synchronous group buffer pool index reads monitor element**

On a DB2 pureScale environment, the number of times an index page was expected to be in the bufferpool, but was instead retrieved from the group bufferpool. This value will be 0 for environments outside of a DB2 pureScale environment.

---

## pool\_sync\_index\_reads - Synchronous buffer pool index reads monitor element

The number of times an index page was expected to be in the bufferpool, but was instead read from disk.

---

## pool\_sync\_xda\_gbp\_reads - Synchronous group buffer pool XDA data reads monitor element

On a DB2 pureScale environment, the number of times an XML page was expected to be in the bufferpool, but was instead retrieved from the group bufferpool. This value will be 0 for environments outside of a DB2 pureScale environment.

---

## pool\_sync\_xda\_reads - Synchronous buffer pool XDA data reads monitor element

The number of times an XML page was expected to be in the bufferpool, but was instead read from disk.

---

## pool\_temp\_data\_l\_reads - Buffer pool temporary data logical reads monitor element

Indicates the number of data pages which have been requested from the buffer pool (logical) for temporary table spaces.

*Table 1321. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 1321. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1322. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1323. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

In conjunction with the **pool\_temp\_data\_p\_reads** element, you can calculate the data page hit ratio for buffer pools located in temporary table spaces.

For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

---

## pool\_temp\_data\_p\_reads - Buffer pool temporary data physical reads monitor element

Indicates the number of data pages read in from the table space containers (physical) for temporary table spaces.

*Table 1324. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 1324. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1325. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1326. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE   |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

In conjunction with the **pool\_temp\_data\_1\_reads** element, you can calculate the data page hit ratio for buffer pools located in temporary table spaces. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467

## pool\_temp\_index\_l\_reads - Buffer pool temporary index logical reads monitor element

Indicates the number of index pages which have been requested from the buffer pool (logical) for temporary table spaces.

*Table 1327. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1328. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1329. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

Use this element, in conjunction with the **pool\_temp\_index\_p\_reads** element, to calculate the index page hit ratio for buffer pools located in temporary table spaces. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

---

## pool\_temp\_index\_p\_reads - Buffer pool temporary index physical reads monitor element

Indicates the number of index pages read in from the table space containers (physical) for temporary table spaces.

Table 1330. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1330. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE            |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1331. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

*Table 1332. Event Monitoring Information*

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE   |

## Usage

Use this element, in conjunction with the **pool\_temp\_index\_1\_reads** element, to calculate the index page hit ratio for buffer pools located in temporary table spaces. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467.

---

## pool\_temp\_xda\_1\_reads - Buffer pool temporary XDA data logical reads monitor element

Indicates the number of pages for XML storage object (XDA) data which have been requested from the buffer pool (logical) for temporary table spaces.

*Table 1333. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1333. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1334. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1335. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Database   | event_db  | Always collected      |

Table 1335. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch         |
|---------------|--|------------------------|
| Tablespaces   | event_tablespace                           | Always collected       |
| Connection    | event_conn                                 | Always collected       |
| Statement     | event_stmt                                 | Always collected       |
| Activities    | event_activity                             | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE  |

## Usage

You can use the **pool\_temp\_xda\_l\_reads** monitor element in conjunction with **pool\_temp\_xda\_p\_reads**, **pool\_temp\_data\_l\_reads**, and **pool\_temp\_data\_p\_reads** monitor elements to calculate the data page hit ratio for buffer pools located in temporary table spaces by using the following formula:

$$1 - \left( \frac{\text{pool\_temp\_data\_p\_reads} + \text{pool\_temp\_xda\_p\_reads}}{\text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads}} \right)$$

## pool\_temp\_xda\_p\_reads - Buffer pool temporary XDA data physical reads monitor element

Indicates the number of pages for XML storage object (XDA) data read in from the table space containers (physical) for temporary table spaces.

Table 1336. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 1336. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1337. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1338. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

Table 1338. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                    | Monitor Switch       |
|--------------|--|----------------------|
| Unit of work | Reported in the system_metrics document. | REQUEST METRICS BASE |

## Usage

You can use the **pool\_temp\_xda\_p\_reads** monitor element in conjunction with **pool\_temp\_xda\_l\_reads**, **pool\_temp\_data\_l\_reads**, and **pool\_temp\_data\_p\_reads** monitor elements to calculate the data page hit ratio for buffer pools located in temporary table spaces by using the following formula:

$$1 - ((\text{pool\_temp\_data\_p\_reads} + \text{pool\_temp\_xda\_p\_reads}) / (\text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads}))$$

## pool\_watermark - Memory Pool Watermark

The largest size of a memory pool since its creation. The value is given in bytes.

### Element identifier

pool\_watermark

### Element type

Information

Table 1339. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 1340. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbmemuse        | Always collected |
| Connection | event_connmemuse      | Always collected |

**Usage** On continuously running systems, you can use the *pool\_watermark* and *pool\_config\_size* elements together to predict potential memory problems.

For example, take a snapshot at regular intervals (for example, daily), and examine the *pool\_watermark* and *pool\_config\_size* values. If you observe that the value of *pool\_watermark* is becoming increasingly close to *pool\_config\_size* (a premature indication of potential future memory-related problems), this may indicate that you should increase the size of the memory pool.

## pool\_write\_time - Total buffer pool physical write time monitor element

Provides the total amount of time spent physically writing data or index pages from the buffer pool to disk. Elapsed time is given in milliseconds.

Table 1341. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1341. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1342. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1343. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Tablespaces   | event_tablespace                                      | Always collected      |
| Connection    | event_conn  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element with **pool\_data\_writes** and **pool\_index\_writes** monitor elements to calculate the average page-write time. This average is important since it may indicate the presence of an I/O wait, which in turn may indicate that you should be moving data to a different device.

At the database and table space levels, this element includes the value of the **pool\_async\_write\_time** monitor element.

## pool\_xda\_gbp\_indep\_pages \_found\_in\_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element

The number of group buffer pool (GBP) independent XML storage object (XDA) data pages found in a local buffer pool (LBP) by an agent.

*Table 1344. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries (reported in DETAILS XML document)         | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1345. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache_metrics      | ACTIVITY METRICS BASE |
| Unit of Work  | uow_metrics           | REQUEST METRICS BASE  |

## pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element

The number of times a request for a data page for an XML storage object (XDA) was made from the group buffer pool due to the page being marked invalid in the local buffer pool.

Table 1346. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |

Table 1346. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1347. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was attempted to be read from the group buffer pool because the page was either invalid or not present in the local buffer pool.

Table 1348. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1348. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE            |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE             |

Table 1349. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

Table 1349. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was read into the local buffer pool from disk because it was not found in the group buffer pool.

Table 1350. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |

Table 1350. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1351. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_xda\_l\_reads - Buffer pool XDA data logical reads monitor element

Indicates the number of data pages for XML storage objects (XDAs) which have been requested from the buffer pool (logical) for regular and large table spaces.

*Table 1352. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_CONTAINER table function - Get table space container metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1353. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

Table 1353. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1354. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Unit of work  | Reported in the system_metrics document.              | Always collected       |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

This count includes accesses to data that is:

- Already in the buffer pool when the database manager needs to process the page
- Read into the buffer pool before the database manager can process the page.

Use the **pool\_xda\_l\_reads**, **pool\_xda\_p\_reads**, **pool\_data\_l\_reads**, and **pool\_data\_p\_reads** monitor elements to calculate the data page hit ratio for the buffer pool. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467

For example, the overall buffer pool hit ratio can be calculated as follows:

$$\frac{((\text{pool\_data\_lbp\_pages\_found} + \text{pool\_index\_lbp\_pages\_found} + \text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_data\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}))}{(\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads} + \text{pool\_xda\_l\_reads} + \text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads} + \text{pool\_temp\_index\_l\_reads})} \times 100$$

This calculation takes into account all of the pages (index and data) that are cached by the buffer pool.

Increasing buffer pool size will generally improve the hit ratio, but you will reach a point of diminishing return. Ideally, if you could allocate a buffer pool large enough to store your entire database, then once the system is up and running you would get a hit ratio of 100%. However, this is unrealistic in most cases. The significance of the hit ratio depends on the size of your data, and the way it is accessed. A very large database where data is accessed evenly would have a poor hit ratio. There is little you can do with very large tables. In such case, you would focus your attention on smaller, frequently accessed tables, and on the indexes.

## pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element

The number of times a data page for an XML storage object (XDA) was requested from and found in the local buffer pool.

*Table 1355. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |

Table 1355. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1356. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element

Indicates the number of data pages for XML storage objects (XDAs) read in from the table space containers (physical) for regular and large table spaces.

Table 1357. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 1357. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1358. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Buffer Pool            |
| Table Space    | tablespace            | Buffer Pool            |
| Buffer Pool    | bufferpool            | Buffer Pool            |
| Application    | appl                  | Buffer Pool            |
| Application    | stmt                  | Buffer Pool            |
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 1359. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch         |
|---------------|---|------------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE  |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE   |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE   |
| Database      | event_db  | Always collected       |
| Tablespaces   | event_tablespace                                      | Always collected       |
| Connection    | event_conn  | Always collected       |
| Statement     | event_stmt  | Always collected       |
| Activities    | event_activity  | Buffer Pool, Statement |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE  |

## Usage

Use the **pool\_async\_xda\_reads** and **pool\_xda\_p\_reads** monitor elements to calculate the number of physical reads that were performed synchronously on XML storage object data pages (that is, physical data page reads that were performed by database manager agents on XML data). Use the following formula:

$$\text{pool\_xda\_p\_reads} - \text{pool\_async\_xda\_reads}$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num\_ioservers** configuration parameter.

Use the **pool\_xda\_l\_reads**, **pool\_xda\_p\_reads**, **pool\_data\_l\_reads**, and **pool\_data\_p\_reads** monitor elements to calculate the data page hit ratio for the buffer pool. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467

---

## pool\_xda\_writes - Buffer pool XDA data writes monitor element

Indicates the number of times a buffer pool data page for an XML storage object (XDA) was physically written to disk.

Table 1360. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 1360. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1361. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 1362. Event Monitoring Information

| Event Type | Logical Data Grouping                                    | Monitor Switch        |
|------------|--|-----------------------|
| Activities | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                    | ACTIVITY METRICS BASE |

Table 1362. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Database      | event_db   | Always collected      |
| Tablespaces   | event_tablespace                                     | Always collected      |
| Connection    | event_conn   | Always collected      |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## Usage

This monitor element helps you to assess whether performance may be improved by increasing the number of buffer pool pages available for the database. For databases containing XML data, you should consider the ratio of buffer pool page writes to buffer pool page reads both for XML data (using the **pool\_xda\_writes** and the **pool\_xda\_p\_reads** monitor elements) and for relational data types (using the **pool\_data\_writes** and the **pool\_data\_p\_reads** monitor elements).

Use the **pool\_xda\_l\_reads**, **pool\_xda\_p\_reads**, **pool\_data\_l\_reads**, and **pool\_data\_p\_reads** monitor elements to calculate the data page hit ratio for the buffer pool. For more information, see “Formulas for calculating buffer pool hit ratios” on page 1467

---

## port\_number - Port number monitor element

The TCP/IP port that a member is listening on for client connections.

Table 1363. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| DB_MEMBERS table function                                       | Always collected                 |
| MON_GET_SERVERLIST table function - get member priority details | Always collected                 |

---

## post\_shrthreshold\_hash\_joins - Post threshold hash joins

The total number of hash joins that were throttled back by the sort memory throttling algorithm. A throttled hash join is a hash join that was granted less memory than requested by the sort memory manager.

Table 1364. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

For snapshot monitoring, this counter can be reset.

Table 1365. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

A hash join is throttled back when the memory allocation from the shared sort heap is close to the limit set by database configuration parameter *sheapthres\_shr*. This throttling will significantly reduce the number of overflows over *sheapthres\_shr* limit in a system that is not properly configured. The data reported in this element only reflects hash joins using memory allocated from the shared sort heap.

## post\_shrthreshold\_sorts - Post shared threshold sorts monitor element

The total number of sorts that were throttled back by the sort memory throttling algorithm. A throttled sort is a sort that was granted less memory than requested by the sort memory manager.

Table 1366. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1366. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1367. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Sort           |

For snapshot monitoring, this counter can be reset.

Table 1368. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

A sort is throttled back when the memory allocation for sorts is close to the limit set by database configuration parameter **sheapthres\_shr**. This throttling will significantly reduce the number of overflows over **sheapthres\_shr** limit in a system that is not properly configured. The data reported in this element only reflects sorts using memory allocated from the shared sort heap.

---

## post\_threshold\_hash\_joins - Hash Join Threshold

The total number of times that a hash join heap request was limited due to concurrent use of shared or private sort heap space.

### Element identifier

post\_threshold\_hash\_joins

### Element type

counter

Table 1369. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** If this value is large (greater than 5% of hash\_join\_overflows), the sort heap threshold should be increased.

---

## post\_threshold\_olap\_funcs - OLAP Function Threshold monitor element

The number of OLAP functions that have requested a sort heap after the sort heap threshold has been exceeded.

*Table 1370. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

For snapshot monitoring, this counter can be reset.

### Usage

Sorts, hash joins, and OLAP functions are examples of operations which use a sort heap. Under normal conditions, the database manager will allocate sort heap using the value specified by the sortheap configuration parameter. If the amount of memory allocated to sort heaps exceeds the sort heap threshold (sheapthres configuration parameter), the database manager will allocate subsequent sort heaps using a value less than that specified by the sortheap configuration parameter.

OLAP functions which start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute.

To improve sort, hash join, OLAP function performance, and overall system performance, modify the sort heap threshold and sort heap size configuration parameters.

If this element's value is high, increase the sort heap threshold (sheapthres).

---

## post\_threshold\_peas - Partial early aggregation threshold monitor element

The number of times that partial early aggregation operations received less memory than requested due to sort heap threshold being exceeded.

*Table 1371. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics                     | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics    | REQUEST METRICS BASE   |

Table 1371. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics         | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                 | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                         | REQUEST METRICS BASE             |

Table 1372. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |

Table 1372. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)      | REQUEST METRICS BASE             |

Table 1373. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Connection    | event_conn  | -                     |
| Statements    | event_stmt  | -                     |
| Transactions  | event_xact  | -                     |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with the **total\_peas** monitor element to determine if partial early aggregation operations are getting sufficient sort heap memory most of the time. If the ratio of the **post\_threshold\_peas** monitor element to the **total\_peas** monitor element is high, your database performance may be sub-optimal. You should consider increasing the sort heap size or the sort heap threshold, or both.

---

## post\_threshold\_peds - Partial early distincts threshold monitor element

The number of times that partial early distinct operations received less memory than requested due to sort heap threshold being exceeded.

Table 1374. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1374. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element  | Collection Level |
|---|------------------|------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS | BASE             |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS  | BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS  | BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS  | BASE             |

Table 1375. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Connection    | event_conn  | -                     |
| Statements    | event_stmt  | -                     |
| Transactions  | event_xact  | -                     |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with the **total\_peds** monitor element to determine if partial early distinct operations are getting sufficient sort heap memory most of the time. If the ratio of the **post\_threshold\_peds** monitor element to the **total\_peds** monitor element is high, your database performance may be sub-optimal. You should consider increasing the sort heap size or the sort heap threshold, or both.

---

## post\_threshold\_sorts - Post threshold sorts monitor element

The number of sorts that have requested heaps after the sort heap threshold has been exceeded.

*Table 1376. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1377. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Sort           |

For snapshot monitoring, this counter can be reset.

Table 1378. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Under normal conditions, the database manager will allocate sort heap using the value specified by the **sortheap** configuration parameter. If the amount of memory allocated to sort heaps exceeds the sort heap threshold (**sheapthres** configuration parameter), the database manager will allocate sort heap using a value less than that specified by the **sortheap** configuration parameter.

Each active sort on the system allocates memory, which may result in sorting taking up too much of the system memory available. Sorts that start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute, but, as a result, the entire system may benefit. By modifying the sort heap threshold and sort heap size configuration parameters, sort operation performance and overall system performance can be improved. If this element's value is high, you can:

- Increase the sort heap threshold (**sheapthres**) or,
- Adjust applications to use fewer or smaller sorts via SQL query changes.

---

## prefetch\_wait\_time - Time waited for prefetch monitor element

The time an application spent waiting for an I/O server (prefetcher) to finish loading pages into the buffer pool. The value is given in milliseconds.

Table 1379. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1379. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1380. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

Table 1381. Event Monitoring Information

| Event Type    | Logical Data Grouping   | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in details_xml document)                   | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics           | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Database      | event_db  | Bufferpool            |
| Connection    | event_db  | Bufferpool            |

**Usage** This element can be used to experiment with changing the number of I/O servers, and I/O server sizes.

## prefetch\_waits - Prefetcher wait count monitor element

The number of times waited for an I/O server (prefetcher) to finish loading pages into the buffer pool.

Table 1382. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |

Table 1382. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1383. Event Monitoring Information

| Event Type    | Logical Data Grouping  | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in details_xml document)<br>event_activitymetrics | ACTIVITY METRICS BASE |
| Package Cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics        | ACTIVITY METRICS BASE |
| Unit of Work  | uow (reported in the metrics.xml document)<br>uow_metrics                  | REQUEST METRICS BASE  |
| Statistics    | event_scstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                       | REQUEST METRICS BASE  |

## prep\_time - Preparation time monitor element

Time in milliseconds required to prepare an SQL statement (if the activity is an SQL statement; otherwise, the value is 0).

Table 1384. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1385. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Activities    | event_activity        | Always collected  |
| Package cache | -                     | COLLECT BASE DATA |

## Usage

The `prep_time` monitor element indicates how much time was spent preparing the SQL statement, if this activity was an SQL statement, when the statement was first introduced to the DB2 package cache. This preparation time is not part of the activity lifetime nor does it represent time spent during a specific invocation of the statement if the statement has already been cached in the package cache before that invocation.

---

## prep\_time\_best - Statement best preparation time monitor element

The shortest amount of time in milliseconds that was required to prepare a specific SQL statement.

Table 1386. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

## Usage

Use this value in conjunction with `prep_time_worst` to identify SQL statements that are expensive to compile.

---

## prep\_time\_worst - Statement worst preparation time monitor element

The longest amount of time in milliseconds that was required to prepare a specific SQL statement.

Table 1387. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

## Usage

Use this value in conjunction with `prep_time_best` to identify SQL statements that are expensive to compile.

---

## prev\_uow\_stop\_time - Previous Unit of Work Completion Timestamp

This is the time the unit of work completed.

### Element identifier

`prev_uow_stop_time`

### Element type

timestamp

Table 1388. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dcs_appl              | Unit of Work, Timestamp |

Table 1389. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Transaction | event_xact            | Always collected |

**Usage** You may use this element with *uow\_stop\_time* to calculate the total elapsed time between COMMIT/ROLLBACK points, and with *uow\_start\_time* to calculate the time spent in the application between units of work. The time of one of the following actions:

- For applications currently within a unit of work, this is the time that the latest unit of work completed.
- For applications not currently within a unit of work (the application has completed a unit of work, but not yet started a new one), this is the stop time of the last unit of work that completed before the one that just completed. The stop time of the one just completed is indicated *uow\_stop\_time*.
- For applications within their first unit of work, this is the database connection request completion time.

---

## priority - Priority value monitor element

Describes the relative capacity of a member to handle work. The higher the value, the more work a client should drive to that member.

Table 1390. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVERLIST table function - get member priority details | Always collected                 |

### Usage Notes

- This monitor element represents the relative load of a member, also known as the weight. For example, if member A has a priority value of 80 and member B has a priority value of 40, this means that member A should receive double the amount of work that is given to member B.
- This value does not represent a percentage.
- The maximum value of this monitor element is 100.

---

## priv\_workspace\_num\_overflows - Private Workspace Overflows

The number of times that the private workspaces overflowed the bounds of its allocated memory.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

Table 1391. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1392. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Use this element with `priv_workspace_size_top` to determine whether the size of the private workspace needs to be increased to avoid overflowing. Overflows of the private workspace may cause performance degradation as well as out of memory errors from the other heaps allocated out of agent private memory.

At the database level, the element reported will be from the same private workspace as that which was reported as having the same Maximum Private Workspace size. At the application level, it is the number of overflows for the workspace of every agent that have serviced the current application.

---

## priv\_workspace\_section\_inserts - Private Workspace Section Inserts

Inserts of SQL sections by an application into the private workspace.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

Table 1393. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1394. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** The working copy of executable sections are stored in the private workspace.

This counter indicates when a copy was not available and had to be inserted. At the database level, it is the cumulative total of all inserts for every application across all private workspaces in the database. At the application level, it is the cumulative total of all inserts for all sections in the private workspace for this application.

In a concentrator environment where agents are being associated with different applications, additional private workspace inserts may be required as a result of a new agent not having the required section available in its private workspace.

---

## priv\_workspace\_section\_lookups - Private Workspace Section Lookups

Lookups of SQL sections by an application in its agents' private workspace.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1395. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1396. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Each application has access to the private workspace of the agent working for it.

This counter indicates how many times the private workspace was accessed in order to locate a specific section for an application. At the database level, it is the cumulative total of all lookups for every application across all private workspaces in the database. At the application level, it is the cumulative total of all lookups for all sections in the private workspace for this application.

You can use this element in conjunction with Private Workspace Section Inserts to tune the size of the private workspace. The size of the private workspace is controlled by the applheapsz configuration parameter.

---

## priv\_workspace\_size\_top - Maximum Private Workspace Size

The largest size reached by the Private Workspace.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1397. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

Table 1398. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Each agent has a private workspace that the application it is servicing has access to. This element indicates the maximum number of bytes required from a private workspace by any agent servicing it. At the database level, it is the maximum number of bytes required of all the private workspaces for all agents attached to the current database. At the application level, it is the maximum size from among all of the agents' private workspaces that have serviced the current application.

When the private workspace overflows, memory is temporarily borrowed from other entities in agent private memory. This can result in memory shortage errors from these entities or possibly performance degradation. You can reduce the chance of overflow by increasing APPLHEAPSZ.

---

## product\_name - Product Name

Details of the version of the DB2 instance that is running.

Table 1399. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

---

## progress\_completed\_units - Completed Progress Work Units

The number of work units for the current phase which have been completed.

Table 1400. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

The value of this element will typically increase as the utility operates. This element will always be less than or equal to *progress\_total\_units* (if both elements are defined).

**Note:**

1. This element might not be included for all utilities.
2. This element is expressed in units displayed by the *progress\_work\_metric* monitor element.

**Usage** Use this element to determine the amount of completed work within a phase. By itself, this element can be used to monitor the activity of a running utility. This element should constantly increase as the utility executes. If the *progress\_completed\_units* fails to increase over a long period of time then the utility might be stalled.

If *progress\_total\_units* is defined, then this element can be used to calculate the percentage of completed work:

percentage complete =  $\text{progress\_completed\_units} / \text{progress\_total\_units} * 100$

---

## progress\_description - Progress Description

Describes the phase of work.

Table 1401. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

Example values for the load utility include:

- DELETE
- LOAD
- REDO

**Usage** Use this element to obtain a general description of a phase.

---

## progress\_list\_attr - Current Progress List Attributes

This element describes how to interpret a list of progress elements.

Table 1402. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress list         | Basic          |

### Usage

The value for this element is one of the following constants:

- `SQLM_ELM_PROGRESS_LIST_ATTR_SERIAL` - The elements in the list are to be interpreted as a set of serial phases meaning that completed work must equal the total work for element  $n$  before the completed work of element  $n+1$  is first updated. This attribute is used to describe progress of a task which consists of a set of serial phases where a phase must fully complete before the next phase begins.
- `SQLM_ELM_PROGRESS_LIST_ATTR_CONCURRENT` - Any element in the progress list can be updated at any time.

Use this element to determine how the elements of a progress list will be updated.

---

## progress\_list\_cur\_seq\_num - Current Progress List Sequence Number

If the utility contains multiple sequential phases, then this element displays the number of the current phase.

Table 1403. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress_list         | Basic          |

**Usage** Use this element to determine the current phase of a multiphase utility. See “progress\_seq\_num - Progress Sequence Number.”

---

## progress\_seq\_num - Progress Sequence Number

Phase number.

**Note:** The phase number displays only for utilities that consist of multiple phases of execution.

Table 1404. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

**Usage** Use this element to determine the order of phases within a multiphase utility. The utility will execute phases serially in order of increasing progress sequence numbers. The current phase of a multiphase utility can be found by matching the *progress\_seq\_num* with the value of *progress\_list\_current\_seq\_num*.

---

## progress\_start\_time - Progress Start Time

A timestamp representing the start of the phase.

Table 1405. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

**Usage** Use this element to determine when a phase started. This element is omitted if the phase has not yet begun.

---

## progress\_total\_units - Total Progress Work Units

Total amount of work to perform in order for the phase to be complete.

Table 1406. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

Some utilities might not be able to quantify the total work so they will continuously update this element. Other utilities might not be able to provide an estimate for the total work so this element might be omitted entirely.

This element is expressed in units displayed by the *progress\_work\_metric* monitor element.

**Usage** Use this element to determine the total amount of work in the phase. Use this element with *progress\_completed\_units* to calculate the percentage of work completed within a phase:

$$\text{percentage complete} = \text{progress\_completed\_units} / \text{progress\_total\_units} * 100$$

---

## progress\_work\_metric - Progress Work Metric

The metric for interpreting the *progress\_total\_units* and *progress\_completed\_units* elements.

Table 1407. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | progress              | Basic          |

Example values include:

- SQLM\_WORK\_METRIC\_BYTES
- SQLM\_WORK\_METRIC\_EXTENTS

**Note:**

1. This element might not be included for all utilities.
2. Values for this element can be found in sqlmon.h

**Usage** Use this element to determine what *progress\_total\_units* and *progress\_completed\_units* use as their reporting metric.

---

## pseudo\_deletes - Pseudo deletes monitor element

The number of keys that have been marked pseudo deleted.

*Table 1408. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## pseudo\_empty\_pages - Pseudo empty pages monitor element

The number of pages that have been identified as pseudo empty. Pseudo empty pages are pages where all the keys have been pseudo deleted.

*Table 1409. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

### Usage

**Note:** This monitor element does not report the current number of pseudo empty pages.

---

## query\_actual\_degree - Actual runtime degree of intrapartition parallelism monitor element

The actual runtime degree of intrapartition parallelism reported at the statement, activity, transaction, or workload level. The state is either enabled or disabled.

*Table 1410. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 1411. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | -                     | Always collected |
| Locking    |                       |                  |

---

## query\_card\_estimate - Query Number of Rows Estimate

An estimate of the number of rows that will be returned by a query.

Table 1412. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |
| Activities     | event_activity        | -              |

**Usage** This estimate by the SQL compiler can be compared with the run time actuals.

This element also returns information for the following SQL statements when you are monitoring DB2 Connect.

- INSERT, UPDATE, and DELETE  
Indicates the number of rows affected.
- PREPARE

Estimate of the number of rows that will be returned. Only collected if the DRDA server is DB2 Database for Linux, UNIX, and Windows, DB2 for VM and VSE, or DB2 for OS/400®.

- FETCH

Set to the number of rows fetched. Only collected if the DRDA server is DB2 for OS/400.

If information is not collected for a DRDA server, then the element is set to zero.

---

## query\_cost\_estimate - Query cost estimate monitor element

Estimated cost for a query, as determined by the SQL compiler. This value is reported in timerons.

Table 1413. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                        | ACTIVITY METRICS BASE            |

Table 1414. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch    |
|----------------|-----------------------|-------------------|
| Application    | stmt                  | Statement         |
| DCS Statement  | dcs_stmt              | Statement         |
| Activities     | event_activity        | -                 |
| Package cache  | -                     | COLLECT BASE DATA |

## Usage

This monitor element allows correlation of actual run time with the compile-time estimates.

This element also returns information for the following SQL statements when you are monitoring DB2 Connect.

- PREPARE  
Represents the relative cost of the prepared SQL statement.
- FETCH  
Contains the length of the row retrieved. Only collected if the DRDA server is DB2 for OS/400.

If information is not collected for a DRDA server, then the element is set to zero.

**Note:** If the DRDA server is DB2 for OS/390<sup>®</sup> and z/OS, this estimate could be higher than 2\*\*32 - 1 (the maximum integer number that can be expressed through an unsigned long variable). In that case, the value returned by the monitor for this element will be 2\*\*32 - 1.

## query\_data\_tag\_list - Estimated query data tag list monitor element

A comma separated list of data tag values that the compiler estimates will be referenced in a statement. A data tag value is in the list if the compiler predicts the statement will access a table whose data table space has a non-zero data tag attribute defined. The list does not contain duplicate values.

Table 1415. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details                         | ACTIVITY METRICS BASE                        |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE                        |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE                        |

Table 1416. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activity        | ACTIVITY METRICS BASE |
| Package cache | -                     | COLLECT BASE DATA     |

## Usage note

If none of the data table spaces accessed by the query have a data tag defined, the list is empty.

---

## queue\_assignments\_total - Queue assignments total monitor element

The number of times any connection or activity was assigned to this threshold queue since the last reset.

*Table 1417. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

*Table 1418. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |

## Usage

This element can be used to determine the number of times any connection or activity was queued in this particular queue in a given period of time determined by the statistics collection interval. This can help to determine the effectiveness of queuing thresholds.

---

## queue\_start\_time - Queue start timestamp monitor element

The date and time the application started waiting in the queue to to obtain a threshold ticket.

*Table 1419. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

## queue\_size\_top - Queue size top monitor element

Highest queue size that has been reached since the last reset.

*Table 1420. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

*Table 1421. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |

## Usage

Use this element to gauge the effectiveness of queuing thresholds and to detect when queuing is excessive.

---

### queue\_time\_total - Queue time total monitor element

Sum of the times spent in the queue for all connections or activities placed in this queue since the last reset. Units are milliseconds.

*Table 1422. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

*Table 1423. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_qstats          | Always collected |

This element is used to gauge the effectiveness of queuing thresholds and to detect when queuing is excessive.

## Usage notes

**queue\_time\_total** is not reset at the end of a statistic collection interval. If **queue\_time\_total** is used over multiple intervals, it can be greater than the product of **wlm\_collect\_int** and **queue\_size\_top**.

---

### queued\_agents - Queued threshold agents monitor element

The total number of agents currently queued in the threshold.

*Table 1424. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Locking    | lock_participants     |                |

---

### quiescer\_agent\_id - Quiescer Agent Identification

Agent ID of the agent holding a quiesce state.

#### Element identifier

quiescer\_agent\_id

#### Element type

information

*Table 1425. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_quiescer   | Basic          |

**Usage** Use this element in conjunction with **quiescer\_auth\_id** to determine who is responsible for quiescing a table space.

---

## quiescer\_auth\_id - Quiescer User Authorization Identification

Authorization ID of the user holding a quiesce state.

**Element identifier**  
quiescer\_auth\_id

**Element type**  
information

*Table 1426. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_quiescer   | Basic          |

**Usage** Use this element to determine who is responsible for quiescing a table space.

---

## quiescer\_obj\_id - Quiescer Object Identification

The object ID of the object that causes a table space to be quiesced.

**Element identifier**  
quiescer\_obj\_id

**Element type**  
information

*Table 1427. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_quiescer   | Basic          |

**Usage** Use this element in conjunction with quiescer\_ts\_id and quiescer\_auth\_id to determine who is responsible for quiescing a table space. The value of this element matches a value from column TABLEID of view SYSCAT.TABLES.

---

## quiescer\_state - Quiescer State

The type of quiesce being done (for example, "SHARE", "INTENT TO UPDATE", or "EXCLUSIVE").

**Element identifier**  
quiescer\_state

**Element type**  
information

*Table 1428. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_quiescer   | Basic          |

**Usage** The value of this element matches the value of constants SQLB\_QUIESCED\_SHARE, SQLB\_QUIESCED\_UPDATE, or SQLB\_QUIESCED\_EXCLUSIVE from sqlutil.h.

---

## quiescer\_ts\_id - Quiescer Table Space Identification

The table space ID of the object that causes a table space to be quiesced.

**Element identifier**

quiescer\_ts\_id

**Element type**

information

*Table 1429. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_quiescer   | Basic          |

**Usage** Use this element in conjunction with quiescer\_obj\_id and quiescer\_auth\_id to determine who is responsible for quiescing a table space. The value of this element matches a value from column TBSpaceID of view SYSCAT.TABLES.

---

## range\_adjustment - Range Adjustment

This value represents the offset into the container array in which a range actually starts.

*Table 1430. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_container\_id - Range Container

An integer that uniquely defines a container within a range.

*Table 1431. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_end\_stripe - End Stripe

This value represents the number of the last stripe in a range.

*Table 1432. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_max\_extent - Maximum Extent in Range

This value represents the maximum extent number that is mapped by a range.

*Table 1433. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_max\_page\_number - Maximum Page in Range

This value represents the maximum page number that is mapped by a range.

*Table 1434. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_num\_containers - Number of Containers in Range

This value represents the number of containers in the current range.

*Table 1435. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_number - Range Number

This value represents the number of a range within the table space map.

*Table 1436. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_offset - Range Offset

The offset from stripe 0 of the beginning of the stripe set to which a range belongs.

*Table 1437. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_start\_stripe - Start Stripe

This value represents the number of the first stripe in a range.

*Table 1438. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## range\_stripe\_set\_number - Stripe Set Number

This value represents the stripe set in which a range resides.

*Table 1439. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_range      | Basic          |

**Usage** This element is applicable only to a DMS table space.

---

## reclaim\_wait\_time - Reclaim wait time monitor element

In a DB2 pureScale environment, this element represents the amount of time spent waiting on page locks, where the lock request caused a page to be reclaimed. The unit of measurement for time is in milliseconds.

*Table 1440. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information                             | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1440. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE            |

Table 1441. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | -   | Always collected      |

## Usage

The time spent waiting for reclaims on space map pages is counted separately and reported in the `spacemappage_reclaim_wait_time` monitor element.

## **reclaimable\_space\_enabled - Reclaimable space enabled indicator monitor element**

If the table space is enabled for reclaimable storage, then this monitor element returns a value of 1. Otherwise, it returns a value of 0.

Table 1442. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

---

## regvar\_collection\_type - Registry variable collection type

Indicates when the registry variable value was collected.

*Table 1443. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | REGVAR                | Always collected |

### Usage

The change history event monitor collected this value as:

- I** The initial value that was captured when the event monitor was activated.
- U** Updated value

---

## regvar\_level - Registry variable level

Indicates the level of the registry variable.

*Table 1444. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | REGVAR                | Always collected |

### Usage

For the change history event monitor, the level of the registry variable is one of:

- E** Environment
- G** Global
- I** Instance-level
- P** Database partition

---

## regvar\_name - Registry variable name

The name of the registry variable.

*Table 1445. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | REGVAR                | Always collected |

### Usage

For the change history event monitor, this element identifies the registry variable that was updated as part of a REGVAR event, or captured at event monitor startup as part of a REGVARVALUES event. These events represent the following:

#### REGVAR

Changing a registry variable value

#### REGVARVALUES

Capturing registry variable values at event monitor startup

---

## regvar\_old\_value - Registry variable old value

The old value for the registry variable.

Table 1446. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | REGVAR                | Always collected |

### Usage

For the change history event monitor, if the registry variable value was not set, this value is an empty string.

---

## regvar\_value - Registry variable value

This is the value for the registry variable.

Table 1447. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | REGVAR                | Always collected |

### Usage

For the change history event monitor, if the value was not set, this value is an empty string.

Only immediate registry variable updates generate REGVAR events.

---

## rej\_curs\_blk - Rejected Block Cursor Requests

The number of times that a request for an I/O block at server was rejected and the request was converted to non-blocked I/O.

**Element identifier**  
rej\_curs\_blk

**Element type**  
counter

Table 1448. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 1449. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Connection | event_conn            | Always collected |

**Usage** If there are many cursors blocking data, the communication heap may become full. When this heap is full, an error is not returned. Instead, no more I/O blocks are allocated for blocking cursors. If cursors are unable to block data, performance can be affected.

If a large number of cursors were unable to perform data blocking, you may be able to improve performance by:

- Increasing the size of the *query\_heap* database manager configuration parameter.

---

## rem\_cons\_in - Remote Connections To Database Manager

The current number of connections initiated from remote clients to the instance of the database manager that is being monitored.

Table 1450. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

Shows the number of connections from remote clients to databases in this instance. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number does not include applications that were initiated from the same instance as the database manager.

When used in conjunction with the local\_cons monitor element, these elements can help you adjust the setting of the **max\_coordagents** and **max\_connections** configuration parameters.

---

## rem\_cons\_in\_exec - Remote Connections Executing in the Database Manager

The number of remote applications that are currently connected to a database and are currently processing a unit of work within the database manager instance being monitored.

Table 1451. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### Usage

This number can help you determine the level of concurrent processing occurring on the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number does not include applications that were initiated from the same instance as the database manager.

When used in conjunction with the local\_cons\_in\_exec monitor element, this element can help you adjust the setting of the **max\_coordagents** configuration parameter.

If **max\_coordagents** is set to AUTOMATIC, then you do not need to make any adjustments. If it is not set to AUTOMATIC and if the sum of rem\_cons\_in\_exec and local\_cons\_in\_exec is close to **max\_coordagents**, you should increase the value of **max\_coordagents**.

---

## remote\_lock\_time - Remote Lock Time

This element contains the aggregate amount of time, in milliseconds, that this data source spends in a remote lock from all applications or a single application running on this federated server instance since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest. The response time is measured as the difference between the time the federated server submits a remote lock to the data source, and the time the federated server releases a remote lock at the data source.

*Table 1452. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

### Usage

Use this element to determine how much actual time is spent at this data source in a remote lock.

---

## remote\_locks - Remote Locks

This element contains a count of the total number of remote locks that the federated server has called at this data source on behalf of any application since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

*Table 1453. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine how many remote locks were made remotely at the data source.

---

## remote\_member - Remote member monitor element

The numeric identifier for the database member to which data was sent or from which data was received, using the fast communication manager (FCM).

*Table 1454. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

## Usage

All the metrics returned by the `MON_GET_FCM_CONNECTION_LIST` table function apply to the FCM connection between the members described in the `member` and `remote_member` monitor elements.

---

### reopt - Reopt bind option monitor element

The REOPT bind option used to precompile this package. Possible values are: NONE, ONCE, and ALWAYS.

*Table 1455. Event Monitoring Information*

| Event Type | Logical Data Grouping       | Monitor Switch |
|------------|-----------------------------|----------------|
| Locking    | lock_participant_activities |                |

---

### reorg\_completion - Reorganization Completion Flag

Table reorganization success indicator, which includes the reclamation of extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table. For partitioned tables, this value indicates the completion status for the data partition.

*Table 1456. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

**Usage** This element will have a value of 0 if a table or data partition reorganize operation is successful. If a table or data partition reorganize operation is unsuccessful, this element will have a value of -1. Success and failure values are defined in `sqlmon.h` as follows:

- Success: `SQLM_REORG_SUCCESS`
- Failure: `SQLM_REORG_FAIL`

In the case of an unsuccessful table reorganization, see the history file for any diagnostic information, including warnings and errors. This data can be accessed by using the `LIST HISTORY` command. For partitioned tables, the completion status is indicated per data partition. If index re-create fails on a partitioned table, the failed status is updated on all data partitions. See the administration notification log for further diagnostic information.

---

### reorg\_current\_counter - Reorganize Progress

A unit of progress that indicates the amount of reorganization that has been completed. The amount of progress this value represents is relative to the value of `reorg_max_counter`, which represents the total amount of table reorganization that is to be done.

*Table 1457. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

## Usage

You can determine the percentage of table reorganization that has been completed using the following formula:

$$\text{table reorg progress} = \text{reorg\_current\_counter} / \text{reorg\_max\_counter} * 100$$

---

## reorg\_end - Table Reorganize End Time

The end time of a table reorganization including a reorganization to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table. For partitioned tables, this time indicates the end time for each data partition reorganization.

*Table 1458. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_index\_id - Index Used to Reorganize the Table

The index being used to reorganize the table.

*Table 1459. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_long\_tbspc\_id - Table Space Where Long Objects are Reorganized monitor element

The table space in which any long objects (LONG VARCHAR or LOB data) will be reorganized. For partitioned tables, this is the table space in which each partition's LONG VARCHAR and LOB will be reorganized.

*Table 1460. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_max\_counter - Total Amount of Reorganization

A value that indicates the total amount of work to be done in a reorganization. This value includes a reorganization to reclaim extents from multidimensional clustering (MDC) or insert time clustering (ITC) tables. This value can be used with `reorg_current_counter`, which represents the amount of work completed, to determine the progress of a reorganization.

*Table 1461. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_max\_phase - Maximum Reorganize Phase

The maximum number of reorganization phases that occurs during reorganization processing. This number applies to classic (offline) reorganizations and to reorganizations with the RECLAIM EXTENTS option. The range of values is 2 - 4 ([SORT], BUILD, REPLACE,[INDEX\_RECREATE]). The value could also indicate the total amount of work to be done in a reorganization to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table. When such a reorganization occurs, this value is 3 (SCAN, DRAIN, RELEASE).

Table 1462. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_phase - Table reorganization phase monitor element

Indicates the reorganization phase of the table. For partitioned tables, this will also indicate the reorganization phase for each data partition. This applies to offline table reorganizations only.

Table 1463. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

### Usage

For partitioned tables, the reorganization occurs on a data partition by data partition basis. For classic table reorganization, the following phases are possible (phases are listed with their corresponding defines from the `sqlmon.h` file):

- Sort: `SQLM_REORG_SORT`
- Build: `SQLM_REORG_BUILD`
- Replace: `SQLM_REORG_REPLACE`
- Index Recreate: `SQLM_REORG_INDEX_RECREATE`
- Dictionary Build: `SQLM_REORG_DICT_SAMPLE`

For partitioned tables, the Index Recreate phase for partitioned indexes (if any) might be directly entered after the replace phase for that data partition. The **reorg\_phase** element will indicate the Index Recreate phase only after the successful completion of all prior phases on every data partition.

During XDA object compression, the XML data reorganization phase involves reorganizing the XML storage object of the table. The XML dictionary build phase involves attempting to create a compression dictionary for the XML storage object. For XDA object compression, the following two phases are possible:

- XML Reorg: `SQLM_REORG_XML_DATA`
- XML Dictionary Build: `SQLM_REORG_XML_DICT_SAMPLE`

For partitioned tables, where reclamation of extents is being performed, the following phases are possible:

- Scan: `SQLM_REORG_SCAN`
- Drain: `SQLM_REORG_DRAIN`

- Release: SQLM\_REORG\_RELEASE

---

## reorg\_phase\_start - Reorganize Phase Start Time

The start time of a phase of table reorganization or reclaim reorganization. For partitioned tables, this will also indicate the start time of a reorganization phase for each data partition. During the index re-create phase, data groups for all data partitions are updated at the same time for nonpartitioned indexes.

*Table 1464. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_rows\_compressed - Rows Compressed

Number of rows compressed in the table during reorganization.

*Table 1465. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

**Usage** A running count of the number of rows compressed in the table during reorganization. Some records may never be compressed (if the record size is less than the minimum record length).

It is important to note that this row count does not measure the effectiveness of data compression. It only displays the number of records meeting compression criteria.

---

## reorg\_rows\_rejected\_for\_compression - Rows Rejected for Compression

Number of rows that were not compressed during reorganization due to the record length being less than or equal to the minimum record length.

*Table 1466. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

**Usage** A record will not be compressed if it is less than or equal to the minimum record length. The number of rows rejected reflects a running count for these records that fail to meet this compression requirement.

---

## reorg\_start - Table Reorganize Start Time

The start time of a table reorganization including a reorganization to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table. For partitioned tables, this indicates the start time for each data partition reorganization.

*Table 1467. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_status - Table Reorganize Status

The status of an in-place (online) table or a data partition level reorganization. This is not applicable to classic (offline) table reorganizations.

*Table 1468. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

**Usage** An in-place table or data partition reorganization can be in one of the following states (states are listed with their corresponding defines from sqlmon.h):

- Started/Resumed: SQLM\_REORG\_STARTED
- Paused: SQLM\_REORG\_PAUSED
- Stopped: SQLM\_REORG\_STOPPED
- Completed: SQLM\_REORG\_COMPLETED
- Truncate: SQLM\_REORG\_TRUNCATE

An inplace table or data partition reorganization to reclaim extents can be in one of the following states:

- Started: SQLM\_REORG\_STARTED
- Stopped: SQLM\_REORG\_STOPPED
- Completed: SQLM\_REORG\_COMPLETED

---

## reorg\_tbspc\_id - Table Space Where Table or Data partition is Reorganized

The table space in which the table will be reorganized. For partitioned tables, this indicates the table space where each data partition is reorganized.

*Table 1469. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_type - Table Reorganize Attributes

Table reorganize attribute settings.

*Table 1470. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

**Usage** The following are possible attribute settings. Each attribute setting is based upon a bit flag value defined in db2ApiDf.h.

- Allow Write Access: DB2REORG\_ALLOW\_WRITE
- Allow Read Access: DB2REORG\_ALLOW\_READ
- Allow No Access: DB2REORG\_ALLOW\_NONE
- Recluster Via Index Scan: DB2REORG\_INDEXSCAN

- Reorg Long Field LOB Data: DB2REORG\_LONGLOB
- No Table Truncation: DB2REORG\_NOTRUNCATE\_ONLINE
- Replace Compression Dictionary: DB2REORG\_RESET\_DICTIONARY
- Keep Compression Dictionary: DB2REORG\_KEEP\_DICTIONARY
- Reclaim Extents: DB2REORG\_RECLAIM\_EXTS

In addition to the preceding attribute settings, the following attributes are listed in the CLP output of the GET SNAPSHOT FOR TABLES command. These attribute settings are based on the values of other attribute settings or table reorganize monitor elements.

- Reclustering: If the value of the reorg\_index\_id monitor element is non-zero, then the table reorganize operation has this attribute.
- Reclaiming: If the value of the reorg\_index\_id monitor element is zero, then the table reorganize operation has this attribute.
- Inplace Table Reorg: If the reorg\_status monitor element has a value that is not null, then the in-place (online) reorganization method is in use.
- Table Reorg: If the reorg\_phase monitor element has a value that is not null, then the classic (offline) reorganization method is in use.
- Recluster Via Table Scan: If the DB2REORG\_INDEXSCAN flag is not set, then the table reorganize operation has this attribute.
- Reorg Data Only: If the DB2REORG\_LONGLOB flag is not set, then the table reorganize operation has this attribute.

---

## reorg\_xml\_regions\_compressed – XML regions compressed monitor element

Number of XML regions that were compressed during the table reorganization process.

*Table 1471. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## reorg\_xml\_regions\_rejected\_for\_compression – XML regions rejected for compression monitor element

Number of XML regions that were not compressed during the table reorganization process.

*Table 1472. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

---

## req\_agent\_tid - Thread identifier for agent waiting to acquire lock monitor element

Thread identifier of the agent or system entity that is waiting to acquire the lock.

*Table 1473. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

---

## req\_application\_handle - Identifier for application waiting to acquire lock monitor element

System-wide unique ID for the application that is waiting to acquire the lock.

*Table 1474. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

---

## req\_executable\_id - Identifier for statement section waiting to acquire lock monitor element

The binary token generated on the data server that uniquely identifies the SQL statement section which is waiting to acquire a lock. For non-SQL activities, a 0-length string value is returned.

*Table 1475. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

---

## req\_member - Member of application waiting to acquire lock monitor element

Database member where the application waiting to acquire this lock is located.

*Table 1476. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

---

## request\_exec\_time\_avg - Request execution time average monitor element

Arithmetic mean of the execution times for requests associated with this service subclass since the last reset. If the internally tracked average has overflowed, the value -2 is returned. This monitor element returns -1 when COLLECT AGGREGATE REQUEST DATA for the service subclass is set to NONE. Units are milliseconds.

When you remap activities between service subclasses with a REMAP ACTIVITY action, the request\_exec\_time\_avg mean counts the partial request in each subclass involved in remapping.

Table 1477. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | COLLECT AGGREGATE REQUEST DATA   |

Table 1478. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |

### Usage

Use this statistic to quickly understand the average amount of time that is spent processing each request on a member in this service subclass.

This average can also be used to determine whether or not the histogram template used for the request execution time histogram is appropriate. Compute the average request execution time from the request execution time histogram. Compare the computed average with this monitor element. If the computed average deviates from the true average reported by this monitor element, consider altering the histogram template for the request execution time histogram, using a set of bin values that are more appropriate for your data.

---

## rf\_log\_num - Log being rolled forward monitor element

The log being processed in the rollforward operation.

Table 1479. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | rollforward           | Basic          |

### Usage

If a rollforward is in progress, this element identifies the log involved. In a DB2 pureScale environment, the **rf\_log\_num** monitor element identifies the log file from each log stream that is currently involved in the rollforward operation.

---

## rf\_status - Log Phase

The status of the recovery.

**Element identifier**  
rf\_status

**Element type**  
information

*Table 1480. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | rollforward           | Basic          |

**Usage** This element indicates the progression of a recovery. It indicates if the recovery is in an undo (rollback) or redo (rollforward) phase.

---

## rf\_timestamp - Rollforward Timestamp

The timestamp of the last committed transaction..

**Element identifier**  
rf\_timestamp

**Element type**  
timestamp

*Table 1481. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | rollforward           | Timestamp      |

**Usage** If a rollforward is in progress, this is the timestamp of the last committed transaction processed by rollforward recovery. This is an indicator of how far the rollforward operation has progressed.

---

## rf\_type - Rollforward Type

The type of rollforward in progress.

**Element identifier**  
rf\_type

**Element type**  
information

*Table 1482. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | rollforward           | Basic          |

**Usage** An indicator of whether recovery is happening at a database or table space level.

---

## rollback\_sql\_stmts - Rollback Statements Attempted

The total number of SQL ROLLBACK statements that have been attempted.

**Element identifier**  
rollback\_sql\_stmts

**Element type**  
counter

Table 1483. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Database        | dbase                 | Basic          |
| Database        | dbase_remote          | Basic          |
| Application     | appl                  | Basic          |
| Application     | appl_remote           | Basic          |
| DCS Database    | dc_s_dbase            | Basic          |
| DCS Application | dc_s_appl             | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1484. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** A rollback can result from an application request, a deadlock, or an error situation. This element **only** counts the number of rollback statements issued from applications.

At the application level, this element can help you determine the level of database activity for the application and the amount of conflict with other applications. At the database level, it can help you determine the amount of activity in the database and the amount of conflict between applications on the database.

**Note:** You should try to minimize the number of rollbacks, since higher rollback activity results in lower throughput for the database.

It may also be used to calculate the total number of units of work, by calculating the sum of the following expression:

```
commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

---

## rolled\_back\_agent\_id - Rolled Back Agent

Agent that was rolled back when a deadlock occurred.

**Element identifier**  
rolled\_back\_agent\_id

**Element type**  
information

Table 1485. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Deadlocks  | event_deadlock        | Always collected |

**Usage** A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

---

## rolled\_back\_appl\_id - Rolled Back Application

Application id that was rolled back when a deadlock occurred.

**Element identifier**

rolled\_back\_appl\_id

**Element type**

information

Table 1486. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Deadlocks  | event_deadlock        | Always collected |

**Usage** A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

---

## rolled\_back\_participant\_no - Rolled back application participant monitor element

The participant number identifying the rolled back application.

Table 1487. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Deadlocks <sup>1</sup> | event_deadlock        | Always collected |

**1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

**Usage**

A system administrator can use this information to determine which application did not complete its updates, and determine which application should be started.

---

## rolled\_back\_sequence\_no - Rolled Back Sequence Number

The sequence number of the application that was rolled back when a deadlock occurred.

**Element identifier**

rolled\_back\_sequence\_no

**Element type**  
information

*Table 1488. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Deadlocks  | event_deadlock        | Always collected |

**Usage** A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

---

## root\_node\_splits - Root node splits monitor element

Number of times the root node of the index was split during an insert operation.

*Table 1489. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX table function - Get index metrics | ACTIVITY METRICS BASE            |

---

## routine\_id - Routine ID monitor element

A unique routine identifier. This monitor element returns 0 if the activity is not part of a routine.

*Table 1490. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in the DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function  | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries                                | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                           | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                            | ACTIVITY METRICS BASE            |

*Table 1491. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitystmt    | Always collected      |
| Unit of work  | uow_package_list      | Always collected      |
| Package cache | pkgcache_metrics      | ACTIVITY METRICS BASE |

## Usage

The value of this element matches a value from column ROUTINEID of view SYSCAT.ROUTINES. When the activity is part of an SQL PL routine that you declare in another SQL PL routine, the value of this element is the ROUTINEID of the outer routine.

---

## rows\_deleted - Rows deleted monitor element

This is the number of row deletions attempted.

*Table 1492. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

*Table 1493. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Application    | appl                  | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1494. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to gain insight into the current level of activity within the database.

This count does not include the attempts counted in the **int\_rows\_deleted** monitor element.

---

## rows\_fetched - Rows fetched monitor element

The number of rows read from the table.

This monitor element is an alias of the **rows\_read** monitor element.

**Note:** This monitor element reports only the values for the member for which this information is recorded. In multimember database environments, these values might not reflect the correct totals for the whole activity.

Table 1495. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | Statement      |

## Usage

See the **rows\_read** monitor element for details.

---

## rows\_inserted - Rows inserted monitor element

The number of row insertions attempted.

Table 1496. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

Table 1497. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Application    | appl                  | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1498. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to gain insight into the current level of activity within the database.

In a federated system, multiple rows can be inserted, per INSERT statement, because the federated server can push INSERT FROM SUBSELECT to the data source, when appropriate.

This count does not include the attempts counted in the **int\_rows\_inserted** monitor element.

---

## rows\_modified - Rows modified monitor element

The number of rows inserted, updated, or deleted.

This monitor element is an alias of the **rows\_written** monitor element.

*Table 1499. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1500. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 1500. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |
| Activities    | event_activity                             | Statement             |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

See the **rows\_written** monitor element for details.

---

## rows\_read - Rows read monitor element

The number of rows read from the table.

Table 1501. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - Get table metrics   | ACTIVITY METRICS BASE  |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1501. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)      | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_OCCURRENCE<br>_ACTIVITIES table function - Return a list of<br>activities                            | REQUEST METRICS BASE             |

Table 1502. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Table          | table                 | Table          |
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Application    | subsection            | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

Table 1503. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Connection    | event_conn   | Always collected      |
| Tables        | event_table  | Always collected      |
| Statements    | event_stmt   | Always collected      |
| Transactions  | event_xact   | Always collected      |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

This element helps you identify tables with heavy usage for which you may want to create additional indexes. To avoid the maintenance of unnecessary indexes, use the SQL EXPLAIN statement to determine if the package uses an index.

This count is *not* the number of rows that were returned to the calling application. Rather, it is the number of rows that had to be read in order to return the result set. For example, the following statement returns one row to the application, but many rows are read to determine the average salary:

```
SELECT AVG(SALARY) FROM USERID.EMPLOYEE
```

This count includes the value in the **overflow\_accesses** monitor element. Additionally, this count does not include any index accesses. That is, if an access plan uses index access only and the table is not touched to look at the actual row, then the value of the **rows\_read** monitor element is not incremented.

---

## rows\_returned - Rows returned monitor element

The number of rows that have been selected and returned to the application. This element has a value of 0 for partial activity records (for example, if an activity is collected while it is still executing or when a full activity record could not be written to the event monitor due to memory limitations).

This monitor element is an alias of the **fetch\_count** monitor element.

*Table 1504. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1504. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element | Collection Level |
|---|-----------------|------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)            | REQUEST METRICS | BASE             |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                                   | REQUEST METRICS | BASE             |

Table 1505. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Activities    | event_activity  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

This element can be used to help determine thresholds for rows returned to the application or can be used to verify that such a threshold is configured correctly and doing its job.

---

### rows\_returned\_top - Actual rows returned top monitor element

The high watermark for the actual rows returned of DML activities at all nesting levels in a service class or work class. For service classes, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE. For work classes, this monitor element returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action, only the rows\_returned\_top high watermark of the service subclass where an activity completes is updated. High watermarks of service subclasses an activity is mapped to but does not complete in are unaffected.

Table 1506. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_wlstats         | -              |

## Usage

Use this element to know the highest DML activity actual rows returned reached on a member for a service class, workload, or work class in the time interval collected.

---

## rows\_selected - Rows Selected

This is the number of rows that have been selected and returned to the application.

Table 1507. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Application    | appl                  | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1508. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to gain insight into the current level of activity within the database.

This element does not include a count of rows read for actions such as COUNT(\*) or joins.

For a federated system, you can compute the average time to return a row to the federated server from the data source:

$$\text{average time} = \text{rows returned} / \text{aggregate query response time}$$

You can use these results to modify CPU speed or communication speed parameters in SYSCAT.SERVERS. Modifying these parameters can impact whether the optimizer does or does not send requests to the data source.

**Note:** This element is collected at the dcs\_dbase and dcs\_appl snapshot monitor logical data groups if the gateway being monitored is at DB2 database version 7.2 or lower.

---

## rows\_updated - Rows updated monitor element

This is the number of row updates attempted.

*Table 1509. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE table function - Get table metrics                                      | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | DATA OBJECT METRICS EXTENDED     |

*Table 1510. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Application    | appl                  | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1511. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to gain insight into the current level of activity within the database.

This value does not include updates counted in the **int\_rows\_updated** monitor element. However, rows that are updated by more than one update statement are counted for each update.

---

## rows\_written - Rows Written

This is the number of rows changed (inserted, deleted or updated) in the table.

*Table 1512. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Application    | subsection            | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

Table 1513. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Connection   | event_conn            | Always collected |
| Tables       | event_table           | Always collected |
| Statements   | event_stmt            | Always collected |
| Transactions | event_xact            | Always collected |

**Usage** A high value for table-level information indicates there is heavy usage of the table and you may want to use the Run Statistics (RUNSTATS) utility to maintain efficiency of the packages used for this table.

For application-connections and statements, this element includes the number of rows inserted, updated, and deleted in temporary tables.

At the application, transaction, and statement levels, this element can be useful for analyzing the relative activity levels, and for identifying candidates for tuning.

## rqsts\_completed\_total - Total requests completed monitor element

The total number of requests executed, including both application and internal requests. For service subclasses, this monitor element is only updated where the request completes. If the request moved between different service subclasses, it is not counted twice.

Table 1514. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1515. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## savepoint\_id - Savepoint ID

The ID of the savepoint set within a unit of work.

Table 1516. Event Monitoring Information

| Event Type     | Logical Data Grouping        | Monitor Switch   |
|----------------|------------------------------|------------------|
| Change History | DDLSTMTEXEC<br>TXNCOMPLETION | Always collected |

## sc\_work\_action\_set\_id - Service class work action set ID monitor element

If this activity has been categorized into a work class of service class scope, this monitor element displays the ID of the work action set associated with the work class set to which the work class belongs. Otherwise, this monitor element displays the value of 0.

Table 1517. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| WLM_GET_ACTIVITY_DETAILS_COMPLETE table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

Table 1518. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

## Usage

This element can be used with the **sc\_work\_class\_id** element to uniquely identify the service class work class of the activity, if one exists.

---

## sc\_work\_class\_id - Service class work class ID monitor element

If this activity has been categorized into a work class of service class scope, this monitor element displays the ID of the work class assigned to this activity. Otherwise, this monitor element displays the value of 0.

Table 1519. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| WLM_GET_ACTIVITY_DETAILS_COMPLETE table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

Table 1520. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

This element can be used with the **sc\_work\_action\_set\_id** element to uniquely identify the service class work class of the activity, if one exists.

---

## sec\_log\_used\_top - Maximum Secondary Log Space Used

The maximum amount of secondary log space used (in bytes).

Table 1521. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

Table 1522. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 1523. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You may use this element in conjunction with *sec\_logs\_allocated* and *tot\_log\_used\_top* to show your current dependency on secondary logs. If this value is high, you may need larger log files, or more primary log files, or more frequent COMMIT statements within your application.

As a result, you may need to adjust the following configuration parameters:

- logfilsiz
- logprimary
- logsecond
- logarchmeth1

The value will be zero if the database does not have any secondary log files. This would be the case if there were none defined.

**Note:** While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

---

## sec\_logs\_allocated - Secondary Logs Allocated Currently

The total number of secondary log files that are currently being used for the database.

Table 1524. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

Table 1525. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** You may use this element in conjunction with *sec\_log\_used\_top* and *tot\_log\_used\_top* to show your current dependency on secondary logs. If this value is consistently high, you may need larger log files, or more primary log files, or more frequent COMMIT statements within your application.

As a result, you may need to adjust the following configuration parameters:

- logfilsiz
- logprimary
- logsecond
- logarchmeth1

---

## section\_actuals - Section actuals monitor element

A binary string generated at the data server containing runtime statistics for a section that was executed. If section capture or actuals collection are not enabled, the value is a 0 length string. For non-SQL activities (for example, LOAD) the value is a 0 length string.

Table 1526. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

The data collected in the **section\_actuals** monitor element or per connection using WLM\_SET\_CONN\_ENV is used when a section explain is performed using the EXPLAIN\_FROM\_ACTIVITY stored procedure. This data is used during EXPLAIN processing to populate the EXPLAIN\_ACTUALS explain table and represents the runtime statistics for the operators in the access plan.

**Note:**

- Section actuals are only available if they have been enabled (set to BASE) using the **section\_actuals** database configuration parameter or if they have been enabled for a particular application using the WLM\_SET\_CONN\_ENV stored procedure. For more information describing the stored procedure, see WLM\_SET\_CONN\_ENV
- The **section\_actuals** setting specified by the WLM\_SET\_CONN\_ENV procedure for an application takes effect immediately.

---

## section\_env - Section environment monitor element

A blob that contains the section for an SQL statement. It is the actual section contents, that is the executable form of the query plan.

*Table 1527. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch        |
|---------------|-----------------------|-----------------------|
| Activities    | event_activitystmt    | Always collected      |
| Package cache | -                     | COLLECT DETAILED DATA |

### Usage

Use this element with the section explain procedures to explain the statement and view the access plan for the statement.

---

## section\_number - Section number monitor element

The internal section number in the package for a static SQL statement.

*Table 1528. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |

*Table 1529. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

*Table 1530. Event Monitoring Information*

| Event Type                          | Logical Data Grouping | Monitor Switch |
|-------------------------------------|-----------------------|----------------|
| Locking                             | -                     | -              |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | -              |
| Statements                          | event_stmt            | -              |

Table 1530. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Activities    | event_activitystmt    | -                 |
| Package cache | -                     | COLLECT BASE DATA |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

For static SQL statements, you can use this element along with **creator**, **package\_version\_id**, and **package\_name** monitor elements to query the SYSCAT.STATEMENTS system catalog table and obtain the static SQL statement text, using the sample query as follows:

```
SELECT SEQNO, SUBSTR(TEXT,1,120)
FROM SYSCAT.STATEMENTS
WHERE PKGNAME = 'package_name' AND
      PKGSCHEMA = 'creator' AND
      VERSION = 'package_version_id' AND
      SECTNO = section_number
ORDER BY SEQNO
```

**Note:** Exercise caution in obtaining static statement text, because this query against the system catalog table could cause lock contention. Whenever possible, only use this query when there is little other activity against the database.

---

## section\_type - Section type indicator monitor element

Indicates whether the SQL statement section is dynamic or static.

Table 1531. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries  | ACTIVITY METRICS BASE            |

Table 1532. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

## Usage

The possible values for this monitor element are:

- D: dynamic
- S: static

---

## select\_sql\_stmts - Select SQL Statements Executed

The number of SQL SELECT statements that were executed.

Table 1533. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Table Space    | tablespace            | Basic          |
| Application    | appl                  | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1534. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to determine the level of database activity at the application or database level.

You can also use the following formula to determine the ratio of SELECT statements to the total statements:

$$\frac{\text{select\_sql\_stmts}}{(\text{static\_sql\_stmts} + \text{dynamic\_sql\_stmts})}$$

This information can be useful for analyzing application activity and throughput.

---

## select\_time - Query Response Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to queries from all applications or a single application running on this federated server instance since the start of the federated server instance or the last reset of the database monitor counters, whichever is latest.

**Note:** Due to query blocking, not all attempts by the federated server to retrieve a row result in communication processing; the request to get the next row can potentially be satisfied from a block of returned rows. As a result, the aggregate query response time does not always indicate processing at the data source, but it usually indicates processing at either the data source or client.

Table 1535. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element to determine how much actual time is spent waiting for data from this data source. This can be useful in capacity planning and tuning the CPU speed and communication rates in SYSCAT.SERVERS. Modifying these parameters can impact whether the optimizer does or does not send requests to the data source.

The response time is measured as the difference in time between the time the federated server requests a row from the data source, and the time the row is available for the federated server to use.

---

## sequence\_no - Sequence number monitor element

This identifier is incremented whenever a unit of work ends (that is, when a COMMIT or ROLLBACK terminates a unit of work). Together, the **appl\_id** and **sequence\_no** uniquely identify a transaction.

*Table 1536. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

*Table 1537. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Connection                            | event_conn            | -              |
| Connections                           | event_connheader      | -              |
| Statements                            | event_stmt            | -              |
| Transactions                          | event_xact            | -              |
| Deadlocks                             | event_dlconn          | -              |
| Deadlocks with Details                | event_detailed_dlconn | -              |
| Deadlocks with Details History        | event_detailed_dlconn | -              |
| Deadlocks with Details History        | event_stmt_history    | -              |
| Deadlocks with Details History Values | event_detailed_dlconn | -              |
| Deadlocks with Details History Values | event_stmt_history    | -              |

---

## sequence\_no\_holding\_lk - Sequence Number Holding Lock

The sequence number of the application that is holding a lock on the object that this application is waiting to obtain.

**Element identifier**

sequence\_no\_holding\_lk

**Element type**

information

Table 1538. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 1539. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Deadlocks              | event_dlconn          | Always collected |
| Deadlocks with Details | event_detailed_dlconn | Always collected |

**Usage** This identifier is used in tandem with `appl_id` to uniquely identify a transaction that is holding a lock on the object that this application is waiting to obtain.

---

## server\_db2\_type - Database Manager Type at Monitored (Server) Node

Identifies the type of database manager being monitored.

Table 1540. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

**Usage** It contains one of the following types of configurations for the database manager:

**API Symbolic Constant**

**Command Line Processor Output**

**sqlf\_nt\_server**

Database server with local and remote clients

**sqlf\_nt\_stand\_req**

Database server with local clients

The API symbolic constants are defined in the include file `sqlutil.h`.

---

## server\_instance\_name - Server Instance Name

The name of the database manager instance for which the snapshot was taken.

**Element identifier**

`server_instance_name`

**Element type**

information

Table 1541. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

Table 1542. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |

**Usage** If more than one instance of the database manager is present on the same system, this data item is used to uniquely identify the instance for which the snapshot call was issued. This information can be useful if you are saving your monitor output in a file or database for later analysis, and you need to differentiate the data from different instances of the database manager.

---

## server\_platform - Server Operating System

The operating system running the database server.

**Element identifier**

server\_platform

**Element type**

information

*Table 1543. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 1544. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** This element can be used for problem determination for remote applications. Values for this field can be found in the header file *sqlmon.h*.

---

## server\_prdid - Server Product/Version ID

The product and version that is running on the server.

*Table 1545. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

*Table 1546. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

**Usage** It is in the form PPPVRRM, where:

**PPP** is SQL

**VV** identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)

**RR** identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)

**M** identifies a 1-character modification level (0-9 or A-Z)

---

## server\_version - Server Version

The version of the server returning the information.

Table 1547. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

### Usage

This field identifies the level of the database server collecting database system monitor information. This allows applications to interpret the data based on the level of the server returning the data. Valid values are:

#### SQLM\_DBMON\_VERSION1

Data was returned by DB2 Version 1

#### SQLM\_DBMON\_VERSION2

Data was returned by DB2 Version 2

#### SQLM\_DBMON\_VERSION5

Data was returned by DB2 Universal Database™ Version 5

#### SQLM\_DBMON\_VERSION5\_2

Data was returned by DB2 Universal Database Version 5.2

#### SQLM\_DBMON\_VERSION6

Data was returned by DB2 Universal Database Version 6

#### SQLM\_DBMON\_VERSION7

Data was returned by DB2 Universal Database Version 7

#### SQLM\_DBMON\_VERSION8

Data was returned by DB2 Universal Database Version 8

#### SQLM\_DBMON\_VERSION9

Data was returned by DB2 Database for Linux, UNIX, and Windows Version 9

#### SQLM\_DBMON\_VERSION9\_5

Data was returned by DB2 Database for Linux, UNIX, and Windows Version 9.5

---

## service\_class\_id - Service class ID monitor element

Unique ID of service subclass. For a unit of work, this ID represents the service subclass ID of the workload with which the connection issuing the unit of work is associated.

Table 1548. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                                     | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics                    | ACTIVITY METRICS BASE            |

Table 1548. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                      | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics     | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics                 | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities | ACTIVITY METRICS BASE            |

Table 1549. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Locking      | -   | Always collected      |
| Unit of work | -   | Always collected      |
| Statistics   | event_histogrambin                                    | Always collected      |
| Statistics   | event_scstats   | Always collected      |

## Usage

The value of this element matches a value from column SERVICECLASSID of view SYSCAT.SERVICECLASSES. Use this element to look up the service subclass name, or link information about a service subclass from different sources. For example, join service class statistics with histogram bin records.

The value of this element is 0 when the following conditions are met:

- The element is reported in an event\_histogrambin logical data group.
- The histogram data is collected for an object that is not a service class.

## service\_level - Service Level

This is the current corrective service level of the DB2 instance.

Table 1550. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

## service\_subclass\_name - Service subclass name monitor element

The name of a service subclass.

Table 1551. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics | ACTIVITY METRICS BASE            |

Table 1551. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics  | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function -<br>Return threshold queue statistics   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table<br>function - list agents running in a service class                                       | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list workload occurrences                                     | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses                                    | ACTIVITY METRICS BASE            |

Table 1552. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Locking      | -  | Always collected     |
| Unit of work | -  | Always collected     |
| Activities   | event_activity                                       | Always collected     |
| Statistics   | event_scstats  | Always collected     |
| Statistics   | event_qstats   | Always collected     |

## Usage

Use this element in conjunction with other activity elements for analysis of the behavior of an activity or with other statistics elements for analysis of a service class or threshold queue.

---

## service\_superclass\_name - Service superclass name monitor element

The name of a service superclass.

Table 1553. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics | ACTIVITY METRICS BASE            |

Table 1553. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics  | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function -<br>Return threshold queue statistics   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table<br>function - list agents running in a service class                                       | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list workload occurrences                                     | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses                                    | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUPERCLASS_STATS<br>table function - Return statistics of service superclasses                                | ACTIVITY METRICS BASE            |

Table 1554. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | -  | Always collected     |
| Activities   | event_activity                                       | Always collected     |
| Statistics   | event_scstats  | Always collected     |
| Statistics   | event_qstats   | Always collected     |

## Usage

Use this element in conjunction with other activity elements for analysis of the behavior of an activity or with other statistics elements for analysis of a service class or threshold queue.

---

## session\_auth\_id - Session authorization ID monitor element

The current authorization ID for the session being used by this application. For monitoring workload management activities, this monitor element describes the session authorization ID under which the activity was injected into the system.

*Table 1555. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)     | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                               | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                              | ACTIVITY METRICS BASE            |

*Table 1556. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Lock           | appl_lock_list        | Basic          |

*Table 1557. Event Monitoring Information*

| Event Type           | Logical Data Grouping | Monitor Element Collection Level |
|----------------------|-----------------------|----------------------------------|
| Unit of work         | -                     | Always collected                 |
| Activities           | event_activity        | Always collected                 |
| Threshold violations | event_activity        | Always collected                 |
| Change history       | changesummary         | Always collected                 |

### Usage

You can use this element to determine what authorization ID is being used to prepare SQL statements, execute SQL statements, or both. This monitor element does not report any session authorization ID values set within executing stored procedures.

---

## shr\_workspace\_num\_overflows - Shared Workspace Overflows

The number of times that shared workspaces overflowed the bounds of their allocated memory.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1558. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1559. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Use this element with `shr_workspace_size_top` to determine whether the size of the Shared Workspaces need to be increased to avoid overflowing. Overflows of Shared Workspaces may cause performance degradation as well as out of memory errors from the other heaps allocated out of application shared memory.

At the database level, the element reported will be from the same shared workspace as that which was reported as having the Maximum Shared Workspace Size. At the application level, it is the number of overflows for the workspace used by the current application.

---

## shr\_workspace\_section\_inserts - Shared Workspace Section Inserts

Number of inserts of SQL sections by applications into shared workspaces.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1560. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1561. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** The working copy of executable sections are stored in shared workspaces. This counter indicates when a copy was not available and had to be inserted.

At the database level, it is the cumulative total of all inserts for every application across all shared workspaces in the database. At the application level, it is the cumulative total of all inserts for all sections in the shared workspace for this application.

---

## shr\_workspace\_section\_lookups - Shared Workspace Section Lookups

Lookups of SQL sections by applications in shared workspaces.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1562. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1563. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Each application has access to a shared workspace where the working copy of executable sections are kept.

This counter indicates how many times shared workspaces were accessed in order to locate a specific section for an application. At the database level, it is the cumulative total of all lookups for every application across all Shared Workspaces in the database. At the application level, it is the cumulative total of all lookups for all sections in the shared workspace for this application.

You can use this element in conjunction with Shared Workspace Section Inserts to tune the size of shared workspaces. The size of the shared workspace is controlled by the `app_ctl_heap_sz` configuration parameter.

---

## shr\_workspace\_size\_top - Maximum Shared Workspace Size

The largest size reached by shared workspaces.

**Note:** This monitor element has been deprecated. Using this monitor element will not generate an error. However, it does not return a valid value. This monitor element is no longer recommended and might be removed in a future release.

*Table 1564. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

Table 1565. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** This element indicates the maximum number of bytes the shared workspaces required for the workload run against the database since it was activated. At the database level, it is the maximum size reached by all of the shared workspaces. At the application level, it is the maximum size of the shared workspace used by the current application.

If a shared workspace overflowed, then this element contains the largest size reached by that shared workspace during the overflow. Check Shared Workspace Overflows to determine if such a condition occurred.

When the shared workspace overflows, memory is temporarily borrowed from other entities in application shared memory. This can result in memory shortage errors from these entities or possibly performance degradation. You can reduce the chance of overflow by increasing APP\_CTL\_HEAP\_SZ.

---

## skipped\_prefetch\_data\_p\_reads - Skipped prefetch data physical reads monitor element

The number of data pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

Table 1566. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be

able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

The **skipped\_prefetch\_\*\_p\_reads** elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the prefetcher was able to retrieve the page, examine the **skipped\_prefetch\_uow\_\*\_p\_reads** monitor elements.

---

## skipped\_prefetch\_index\_p\_reads - Skipped prefetch index physical reads monitor element

The number of index pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

*Table 1567. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the

**db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

The **skipped\_prefetch\_\*\_p\_reads** elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the prefetcher was able to retrieve the page, examine the **skipped\_prefetch\_uow\_\*\_p\_reads** monitor elements.

---

## skipped\_prefetch\_temp\_data\_p\_reads - Skipped prefetch temporary data physical reads monitor element

The number of data pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

*Table 1568. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

The **skipped\_prefetch\_\*\_p\_reads** elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the

prefetcher was able to retrieve the page, examine the **skipped\_prefetch\_uow\_\*\_p\_reads** monitor elements.

---

## skipped\_prefetch\_temp\_index\_p\_reads - Skipped prefetch temporary index physical reads monitor element

The number of index pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

*Table 1569. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

The **skipped\_prefetch\_\*\_p\_reads** elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the prefetcher was able to retrieve the page, examine the **skipped\_prefetch\_uow\_\*\_p\_reads** monitor elements.

---

## skipped\_prefetch\_temp\_xda\_p\_reads - Skipped prefetch temporary XDA data physical reads monitor element

The number of XML storage object (XDA) data pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

Table 1570. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

The **skipped\_prefetch\_\*\_p\_reads** elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the prefetcher was able to retrieve the page, examine the **skipped\_prefetch\_uow\_\*\_p\_reads** monitor elements.

---

## skipped\_prefetch\_uow\_data\_p\_reads - Skipped prefetch unit of work data physical reads monitor element

The number of data pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by an agent in the same unit of work..

Table 1571. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_uow\_\*\_p\_reads** elements tells you the number of pages that were in a prefetch request that were read directly by an agent in the same unit of work that caused the prefetch request to be created. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

---

## skipped\_prefetch\_uow\_index\_p\_reads - Skipped prefetch unit of work index physical reads monitor element

The number of index pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by an agent in the same unit of work.

Table 1572. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

This monitor element, along with the other **skipped\_prefetch\_uow\_\*\_p\_reads** elements tells you the number of pages that were in a prefetch request that were read directly by an agent in the same unit of work that caused the prefetch request to be created. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

---

## skipped\_prefetch\_uow\_temp\_data\_p\_reads - Skipped prefetch unit of work temporary data physical reads monitor element

The number of data pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by an agent in the same unit of work.

*Table 1573. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

This monitor element, along with the other **skipped\_prefetch\_uow\_\*\_p\_reads** elements tells you the number of pages that were in a prefetch request that were read directly by an agent in the same unit of work that caused the prefetch request to be created. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry

variable not being set, which can restrict parallel prefetching within a table space container.

---

## skipped\_prefetch\_uow\_temp\_index\_p\_reads - Skipped prefetch unit of work temporary index physical reads monitor element

The number of index pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by the synchronous transaction.

*Table 1574. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

---

## skipped\_prefetch\_uow\_temp\_xda\_p\_reads - Skipped prefetch unit of work temporary XDA data physical reads monitor element

The number of XML storage object (XDA) data pages for temporary table spaces that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by the synchronous transaction.

*Table 1575. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

---

## skipped\_prefetch\_uow\_xda\_p\_reads - Skipped prefetch unit of work XDA data physical reads monitor element

The number of XML storage object (XDA) data pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool by an agent in the same unit of work.

*Table 1576. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_uow\_\*\_p\_reads** elements tells you the number of pages that were in a prefetch request that were read directly by an agent in the same unit of work that caused the prefetch request to be created. Agents might be forced to read a page directly from disk if a system

has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the **db2\_parallel\_io** registry variable not being set, which can restrict parallel prefetching within a table space container.

---

## skipped\_prefetch\_xda\_p\_reads - Skipped prefetch XDA physical reads monitor element

The number of XML storage object (XDA) data pages that an I/O server (prefetcher) skipped because the pages were already loaded into the buffer pool.

*Table 1577. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

This monitor element, along with the other **skipped\_prefetch\_\*\_p\_reads** elements tells you the number of times a page that was scheduled for retrieval by a prefetcher was not prefetched because it was already in a buffer pool. Pages that are already in a buffer pool might be there for a number of reasons:

- The page is a new page, and is not yet created on disk.
- Another agent might need the same page, and thus it was loaded it into the buffer pool by a different prefetch request. In this case, along with the preceding one, an increase in skipped prefetch requests might not be a problem, because the additional prefetch request that was generated was redundant.
- An agent retrieved them from disk directly before the prefetcher was able to complete the prefetch operation. Agents might be forced to read a page directly from disk if a system has an insufficient number of prefetchers configured, or if there is another type of prefetching bottleneck. . For example, in an OLTP system, where most of the workload is generally transactional in nature, it might be the case that the minimum number of prefetchers is configured by setting the configuration parameter **num\_ioservers** to 1. However, if an operation that uses prefetching, such as a table scan is performed, the single prefetcher might not be able to keep up, and so the agent requests the pages directly. This behavior can cause a degradation in performance, because the application waits on IO that otherwise would have been performed by prefetchers. In this case, consider increasing the number of prefetchers by adjusting the configuration parameter **num\_ioservers**. Other potential causes include having an excessively large prefetch size, which can cause prefetch times that are longer than normal, or the

`db2_parallel_io` registry variable not being set, which can restrict parallel prefetching within a table space container.

The `skipped_prefetch_*_p_reads` elements tell you about all skipped read requests, regardless of the reason the read was skipped. To see how many requests were skipped because an agent from the same unit of work performed a read before the prefetcher was able to retrieve the page, examine the `skipped_prefetch_uow_*_p_reads` monitor elements.

---

## smallest\_log\_avail\_node - Node with Least Available Log Space

This element is only returned for global snapshots and indicates the node with the least amount (in bytes) of available log space.

### Element identifier

`smallest_log_avail_node`

### Element type

information

Table 1578. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** Use this element, in conjunction with `appl_id_oldest_xact`, to ensure that adequate log space is available for the database. In a global snapshot, `appl_id_oldest_xact`, `total_log_used`, and `total_log_available` correspond to the values on this node.

---

## sort\_heap\_allocated - Total Sort Heap Allocated

The total number of allocated pages of sort heap space for all sorts at the level chosen and at the time the snapshot was taken.

Table 1579. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| Database         | dbase                 | Basic          |

**Usage** The amount of memory allocated for each sort may be some or all of the available sort heap size. Sort heap size is the amount of memory available for each sort as defined in the `sortheap` database configuration parameter.

It is possible for a single application to have concurrent sorts active. For example, in some cases a SELECT statement with a subquery can cause concurrent sorts.

Information may be collected at two levels:

- At the database manager level, it represents the sum of sort heap space allocated for all sorts in all active databases in the database manager
- At the database level, it represents the sum of the sort heap space allocated for all sorts in a database.

Normal memory estimates do not include sort heap space. If excessive sorting is occurring, the extra memory used for the sort heap should be added to the base memory requirements for running the database manager.

Generally, the larger the sort heap, the more efficient the sort. Appropriate use of indexes can reduce the amount of sorting required.

You may use the information returned at the database manager level to help you tune the *sheapthres* configuration parameter. If the element value is greater than or equal to *sheapthres*, it means that the sorts are not getting the full sort heap as defined by the *sortheap* parameter.

---

## sort\_heap\_top - Sort private heap high watermark

The private sort memory high watermark, in 4 KB pages, across the database manager.

Table 1580. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

**Usage** This element can be used to determine if the SHEAPTHRES configuration parameter is set to an optimal value. For example, if this watermark approaches or exceeds SHEAPTHRES, it is likely that SHEAPTHRES should be increased. This is because private sorts are given less memory whenever SHEAPTHRES is exceeded, and this can adversely affect system performance.

---

## sort\_overflows - Sort overflows monitor element

The total number of sorts that ran out of sort heap and may have required disk space for temporary storage.

Table 1581. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1581. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)            | REQUEST METRICS BASE             |

Table 1582. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |
| Application    | stmt                  | Basic          |
| Dynamic SQL    | dynsql                | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1583. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db   | Always collected      |
| Connection    | event_conn   | Always collected      |
| Statements    | event_stmt   | Always collected      |
| Activities    | event_activity   | Statement, Sort       |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

At a database or application level, use this element in conjunction with **total\_sorts** to calculate the percentage of sorts that had to overflow to disk. If this percentage is high, you may want adjust the database configuration by increasing the value of **sortheap**.

At a statement level, use this element to identify statements that require large sorts. These statements may benefit from additional tuning to reduce the amount of sorting required.

When a sort overflows, additional processing time is required because the sort will require a merge phase and can potentially require more I/O, if data needs to be written to disk.

This element provides information for one statement, one application, or all applications accessing one database.

---

## sort\_shrheap\_allocated - Sort Share Heap Currently Allocated

Total amount of shared sort memory allocated in the database.

*Table 1584. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** This element can be used to assess the threshold for shared sort memory. If this value is frequently much higher or lower than the current shared sort memory threshold, it is likely that the threshold should be adjusted.

**Note:** The "shared sort memory threshold" is determined by the value of the SHEAPTHRES database manager configuration parameter if the SHEAPTHRES\_SHR database configuration parameter is 0. Otherwise, it is determined by the value of SHEAPTHRES\_SHR.

---

## sort\_shrheap\_top - Sort share heap high watermark

Database-wide shared sort memory high watermark in 4 KB pages.

*Table 1585. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** This element can be used to assess whether or not SHEAPTHRES (or SHEAPTHRES\_SHR) is set to an optimal value. For example, if this high watermark is persistently much lower than the shared sort memory threshold, it is likely that this threshold needs to be decreased, thus freeing memory for other database functions. Conversely, if this high watermark begins to approach the shared sort memory threshold, then this might indicate that this threshold needs to be increased. This is important because the shared sort memory threshold is a hard limit. When the total amount of sort memory reaches this threshold, no more shared sorts can be initiated.

This element, along with the high watermark for private sort memory, can also help users determine if the threshold for shared and private sorts need to be set independently of each other. Normally, if the SHEAPTHRES\_SHR database configuration option has a value of 0, then the shared sort memory threshold is determined by the value of the SHEAPTHRES database manager configuration option. However, if there is a large discrepancy between the private and shared sort memory high watermarks, this might be an indication that the user needs to override

SHEAPTHRES and set SHEAPTHRES\_SHR to a more appropriate value that is based on the shared sort memory high watermark.

**Note:** This element reports the high watermark of sort reservation requests granted by the sort memory controller. Requests that are granted do not always result in a similar level of memory allocation, since they only permit consumers of sort heap to allocate memory as necessary, up to the granted amount, during the processing of an SQL request. It is normal for there to be a discrepancy between the value for this element and the high water mark of the shared sort memory pool (pool\_watermark).

---

## source\_service\_class\_id - Source service class ID monitor element

The ID of the service subclass from which an activity was remapped when the threshold violation record to which this element belongs was generated. This element has a value of zero when the threshold action is anything other than a REMAP ACTIVITY action.

*Table 1586. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

### Usage

Use this element to trace the path of an activity through the service classes to which it was remapped. It can also be used to compute aggregates of how many activities were mapped out of a given service subclass.

---

## sp\_rows\_selected - Rows Returned by Stored Procedures

This element contains the number of rows sent from the data source to the federated server at the start of the federated server instance, or the last reset of the database monitor counters as a result of stored procedure operations for this application.

*Table 1587. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** This element has several uses. You can use it to compute the average number of rows sent to the federated server from the data source, per stored procedure, with the following formula:

$$\begin{aligned} & \text{rows per stored procedure} \\ & = \text{rows returned} \\ & / \# \text{ of stored procedures invoked} \end{aligned}$$

You can also compute the average time to return a row to the federated server from the data source for this application:

$$\text{average time} = \text{aggregate stored proc. response time} / \text{rows returned}$$

---

## sql\_chains - Number of SQL Chains Attempted

Represents the number of SQL statements taking  $n$  data transmissions between the DB2 Connect gateway and the host during statement processing. The range  $n$  is specified by the *num\_transmissions\_group* element.

Table 1588. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| Data Transmission | stmt_transmissions    | Basic          |

For snapshot monitoring, this counter can be reset.

For example, if chaining is on, and if PREP and OPEN statements are chained together and the chain takes a total of two transmissions, *sql\_chains* is reported as "1" and *sql\_stmts* is reported as "2".

If chaining is off, then the *sql\_chains* count equals the *sql\_stmts* count.

**Usage** Use this element to get statistics on how many statements used 2, 3, 4 (and so on) data transmissions during their processing. (At least two data transmissions are necessary to process a statement: a send and a receive.) These statistics can give you a better idea of the database or application activity and network traffic at the database or application levels.

**Note:** The *sql\_stmts* monitor element represents the number of attempts made to send an SQL statement to the server. At the transmission level, all statements within the same cursor count as a single SQL statement.

---

## sql\_req\_id - Request Identifier for SQL Statement

The request identifier for an operation in an SQL statement.

Table 1589. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | -              |

**Usage** This identifier increments with each successive SQL operation processed by the database manager since the first application has connected to the database. Its value is unique across the database and uniquely identifies a statement operation.

---

## sql\_reqs\_since\_commit - SQL Requests Since Last Commit

Number of SQL requests that have been submitted since the last commit.

Table 1590. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

**Usage** You can use this element to monitor the progress of a transaction.

---

## sql\_stmts - Number of SQL Statements Attempted

For data transmission snapshots, this element represents the number of SQL statements taking *n* data transmissions between the DB2 Connect gateway and the host during statement processing. The range *n* is specified by the *num\_transmissions\_group* element.

Table 1591. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcs_dbase             | Basic          |
| DCS Application   | dcs_appl              | Basic          |
| Data Transmission | stmt_transmissions    | Basic          |

For snapshot monitoring, this counter can be reset.

For DCS DATABASE snapshots, this statement count is the number of statements since the database was activated.

For DCS APPLICATION snapshots, this statement count is the number of statements since the connection to the database was established by this application.

**Usage** Use this element to measure database activity at the database or application level. To calculate the SQL statement throughput for a given period, you can divide this element by the elapsed time between two snapshots.

For the data transmission level: Use this element to get statistics on how many statements used 2, 3, 4 (and so on) data transmissions during their processing. (At least 2 data transmissions are necessary to process a statement: a send and a receive.) These statistics can give you a better idea of the database or application activity and network traffic at the database or application levels.

**Note:**

1. The *sql\_stmts* monitor element represents the number of attempts made to send an SQL statement to the server:
  - At the application level and database level, each SQL statement within a cursor is counted separately.
  - At the transmission level, all statements within the same cursor count as a single SQL statement.

---

## sqlca - SQL Communications Area (SQLCA)

The SQLCA data structure that was returned to the application at statement completion.

Table 1592. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | -              |
| Activities | event_activity        | -              |

## Usage

The SQLCA data structure can be used to determine if the statement completed successfully. For information about the content of the SQLCA, see "SQLCA (SQL communications area)" in *SQL Reference Volume 1* or "SQLCA data structure" in *Administrative API Reference*.

---

## sqlrowsread\_threshold\_id - SQL rows read threshold ID monitor element

The ID of the SQLROWSREAD threshold that was applied to the activity.

*Table 1593. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which SQLROWSREAD threshold, if any, was applied to the activity.

---

## sqlrowsread\_threshold\_value - SQL rows read threshold value monitor element

The upper bound of the SQLROWSREAD threshold that was applied to the activity.

*Table 1594. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the SQLROWSREAD threshold applied to the activity, if any.

---

## sqlrowsread\_threshold\_violated - SQL rows read threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the SQLROWSREAD threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 1595. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the SQLROWSREAD threshold that was applied to the activity.

---

### sqlrowsreadinsc\_threshold\_id - SQL rows read in service class threshold ID monitor element

The ID of the SQLROWSREADINSC threshold that was applied to the activity.

*Table 1596. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which SQLROWSREADINSC threshold, if any, was applied to the activity.

---

### sqlrowsreadinsc\_threshold\_value - SQL rows read in service class threshold value monitor element

The upper bound of the SQLROWSREADINSC threshold that was applied to the activity.

*Table 1597. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the SQLROWSREADINSC threshold applied to the activity, if any.

---

### sqlrowsreadinsc\_threshold\_violated - SQL rows read in service class threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the SQLROWSREADINSC threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 1598. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the SQLROWSREADINSC threshold that was applied to the activity.

---

### sqlrowsreturned\_threshold\_id - SQL rows read returned threshold ID monitor element

The ID of the SQLROWSRETURNED threshold that was applied to the activity.

*Table 1599. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand which SQLROWSRETURNED threshold, if any, was applied to the activity.

---

### sqlrowsreturned\_threshold\_value - SQL rows read returned threshold value monitor element

The upper bound of the SQLROWSRETURNED threshold that was applied to the activity.

*Table 1600. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the SQLROWSRETURNED threshold applied to the activity, if any.

---

### sqlrowsreturned\_threshold\_violated - SQL rows read returned threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the SQLROWSRETURNED threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 1601. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the SQLROWSRETURNED threshold that was applied to the activity.

---

### sqltempespace\_threshold\_id - SQL temporary space threshold ID monitor element

The ID of the SQLTEMPSPACE threshold that was applied to the activity.

*Table 1602. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |

## Usage

Use this element to understand which SQLTEMPSPACE threshold, if any, was applied to the activity.

---

### sqltempespace\_threshold\_value - SQL temporary space threshold value monitor element

The upper bound of the SQLTEMPSPACE threshold that was applied to the activity.

*Table 1603. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to understand the value of the SQLTEMPSPACE threshold applied to the activity, if any.

---

### sqltempespace\_threshold\_violated - SQL temporary space threshold violated monitor element

This monitor element returns 'Yes' to indicate that the activity violated the SQLTEMPSPACE threshold. 'No' indicates that the activity has not yet violated the threshold.

*Table 1604. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

## Usage

Use this element to determine if the activity violated the SQLTEMPSPACE threshold that was applied to the activity.

---

### spacemappage\_page\_reclaims\_x - Space map page reclaims exclusive access monitor element

The number of times a page related to a space map page was reclaimed by another member in the DB2 pureScale instance before its planned release. The member that reclaimed the page required exclusive access to the space map page.

*Table 1605. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_x** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_x** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_x** monitor element.

---

### spacemappage\_page\_reclaims\_s - Space map page reclaims shared access monitor element

The number of times a page related to a space map page was reclaimed by another member in the DB2 pureScale instance before its planned release. The member that reclaimed the page required shared access to the space map page.

*Table 1606. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_s** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_s** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_s** monitor element.

---

## **spacemappage\_page\_reclaims\_initiated\_x - Space map page reclaims initiated exclusive access monitor element**

The number of times a page accessed in exclusive mode for a space map page caused the page to be reclaimed from another member.

*Table 1607. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE                   |

### **Usage**

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_initiated\_x** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_initiated\_x** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_initiated\_x** monitor element.

---

## **spacemappage\_page\_reclaims\_initiated\_s - Space map page reclaims initiated shared access monitor element**

The number of times a page accessed in shared mode of a space map page caused the page to be reclaimed from another member.

*Table 1608. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE                   |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_initiated\_s** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_initiated\_s** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_initiated\_s** monitor element.

---

## spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element

In a DB2 pureScale environment, this element represents the amount of time spent waiting on page locks for pages related to internally maintained object space management where the lock request caused a reclaim from another member. The unit of measurement for time is in milliseconds.

*Table 1609. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                               | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information                             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 1609. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1610. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | -   | Always collected      |

## ss\_exec\_time - Subsection Execution Elapsed Time

The time in seconds that it took a subsection to execute.

Table 1611. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

Table 1612. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** Allows you to track the progress of a subsection.

---

## ss\_node\_number - Subsection Node Number

Node where the subsection was executed.

*Table 1613. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 1614. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** Use to correlate each subsection with the database partition where it was executed.

---

## ss\_number - Subsection number monitor element

Identifies the subsection associated with the returned information.

*Table 1615. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_APPL_LOCKWAIT table function<br>- get information about locks for which an application is waiting | ACTIVITY METRICS BASE            |

*Table 1616. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 1617. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statements | event_subsection      | Always collected |

### Usage

This number relates to the subsection number in the access plan that can be obtained with **db2exp1n** command.

---

## ss\_status - Subsection status monitor element

The current status of an executing subsection.

*Table 1618. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

## Usage

The current status values can be:

- executing (SQLM\_SSEXEC in sqlmon.h)
- waiting for a lock
- waiting to receive data on a table queue
- waiting to send data on a table queue

---

## ss\_sys\_cpu\_time - System CPU Time used by Subsection

The total system CPU time (in seconds and microseconds) used by the currently executing statement subsection. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

### Element identifier

ss\_sys\_cpu\_time

### Element type

time

Table 1619. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Timestamp      |

Table 1620. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | Timestamp      |

**Usage** This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

---

## ss\_usr\_cpu\_time - User CPU Time used by Subsection

The total user CPU time (in seconds and microseconds) used by the currently executing statement subsection. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

### Element identifier

ss\_usr\_cpu\_time

### Element type

time

Table 1621. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Timestamp      |

Table 1622. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | Timestamp      |

**Usage** This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

---

## ssl\_port\_number - SSL port number monitor element

The SSL TCP/IP port that a member is listening on for client connections.

*Table 1623. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVERLIST table function - get member priority details | Always collected                 |

---

## start\_event\_id - Start event ID

Unique identifier of the corresponding UTILSTART or UTILSTARTPROC event.

*Table 1624. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTOP              | Always collected |

### Usage

For the change history event monitor, unique identifier of the corresponding starting of a utility event (UTILSTART or UTILSTARTPROC). Use this element with the START\_EVENT\_TIMESTAMP and member elements to associate the stop record with the corresponding start record.

---

## start\_event\_timestamp - Start event timestamp

Time of the corresponding UTILSTART or UTILSTARTPROC event.

*Table 1625. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTOP              | Always collected |

### Usage

For the change history event monitor, use with the START\_EVENT\_ID and member elements to associate the stop record with the corresponding start record.

---

## start\_time - Event Start Time

The date and time of unit of work start, statement start, or deadlock detection. This element, in the event\_start API structure indicates the start of the event monitor.

Table 1626. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Database               | event_start           | Timestamp      |
| Statements             | event_stmt            | Timestamp      |
| Deadlocks              | event_deadlock        | Timestamp      |
| Deadlocks              | event_dlconn          | Timestamp      |
| Deadlocks with Details | event_detailed_dlconn | Timestamp      |

**Usage** You can use this element to correlate the deadlock connection records to the deadlock event record, and in conjunction with *stop\_time* to calculate the elapsed statement or transaction execution time.

**Note:** When the Timestamp switch is OFF, this element reports "0".

---

## static\_sql\_stmts - Static SQL Statements Attempted

The number of static SQL statements that were attempted.

Table 1627. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1628. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to calculate the total number of successful SQL statements at the database or application level:

```
dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= throughput during monitoring period
```

---

## statistics\_timestamp - Statistics timestamp monitor element

The time at which this statistics record was generated.

Table 1629. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wlstats         | -              |

Table 1629. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wcstats         | -              |
| Statistics | event_qstats          | -              |
| Statistics | event_histogrambin    | -              |

## Usage

Use this element to determine when this statistics record was generated.

Use this element along with the **last\_wlm\_reset** element to identify the time interval over which the statistics in this statistics record were generated.

This monitor element can also be used to group together all statistics records that were generated for the same collection interval.

---

## stats\_cache\_size – Size of statistics cache monitor element

The current size of the statistics cache, in bytes, which is used in a catalog partition to cache statistics information generated by real-time statistics gathering.

**Note:** Since the statistics cache resides in the catalog partition, only the snapshot taken at the catalog partition will report the statistics cache size. Snapshots taken at other partitions will report the value of zero instead. When taking a global snapshot, the values reported by all the database partitions are aggregated together.

Table 1630. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

Table 1631. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

Table 1632. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

## Usage

Use this element to determine the size of the current statistics cache. This value changes frequently. In order to evaluate system usage, take the snapshot at specific intervals over an extended period of time. Use this element to adjust the value of

the `catalogcache_sz` configuration parameter.

---

## stats\_fabricate\_time – Total time spent on statistics fabrication activities monitor element

The total time spent on statistics fabrications by real-time statistics gathering, in milliseconds. Statistics fabrication is the statistics collection activity needed to generate statistics during query compilation. If this monitor element is collected at the database level, it represents the total time spent on real-time statistics gathering activities for all the applications running on the database. If it is collected at the statement level, it represents the time spent on the latest real-time statistics gathering activities for the statement. The times reported by all the database partitions are aggregated together.

Table 1633. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

Table 1634. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this element can be reset.

Table 1635. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Statement  | event_stmt            | Always collected |

### Usage

Use this element along with `stats_fabrications` to evaluate the performance impact of real-time statistics gathering at the database level. For snapshot monitor for dynamic SQL, you can use this element along with `total_exec_time` and `num_executions` to evaluate the impact of statistics fabrications. For the statement event monitor, you can combine this element with `stmt_start` and `stmt_stop` for further evaluation of real-time statistics gathering impact.

---

## stats\_fabrications – Total number of statistics fabrications monitor elements

The total number of statistics fabrications performed by real-time statistics during query compilation for all the database applications. Rather than obtaining statistics by scanning data stored in a table or an index, statistics are fabricated based on metadata maintained by the index and data manager. Values reported by all the database partitions are aggregated together.

Table 1636. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

Table 1637. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

Table 1638. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

### Usage

Use this element to determine the frequency of statistics fabrications in the database. This value changes frequently. In order to get a better overview of the system usage, take the snapshot at specific intervals over an extended period of time. When used in conjunction with **stats\_fabricate\_time**, this element can help you evaluate the impact of statistics fabrications.

---

## status\_change\_time - Application Status Change Time

The date and time the application entered its current status.

### Element identifier

status\_change\_time

### Element type

timestamp

Table 1639. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl_id_info          | Unit of Work, Timestamp |
| Lock            | appl_lock_list        | Unit of Work, Timestamp |
| DCS Application | dcs_appl_info         | Unit of Work, Timestamp |

**Usage** This element allows you to determine how long an application has been in its current status. If it has been in the same status for a long period of time, this may indicate that it has a problem.

---

## stmt\_elapsed\_time - Most Recent Statement Elapsed Time

The elapsed execution time of the most recently completed statement.

*Table 1640. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | stmt                  | Statement, Timestamp |
| DCS Statement  | dcs_stmt              | Statement, Timestamp |

### Usage

Use this element as an indicator of the time it takes for a statement to complete.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## stmt\_exec\_time - Statement execution time monitor element

The total time spent executing this statement by all agents on this member. The value is given in milliseconds.

*Table 1641. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based<br>component times                           | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics                             | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW -<br>Get formatted row-based combined hierarchy<br>wait and processing times         | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries             | ACTIVITY METRICS BASE  |

Table 1642. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## stmt\_first\_use\_time - Statement first use timestamp monitor element

This element shows the first time the statement entry was processed. For cursor operations, **stmt\_first\_use\_time** shows when the cursor was opened. At application coordination nodes, this value reflects the application requests; at non-coordinator nodes, this value reflects when requests were received from the originating node.

Table 1643. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--|-----------------------|----------------|
| Locking  | -                     | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history    | timestamp      |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history    | timestamp      |
| Activities   | event_activitystmt    | timestamp      |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

Use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

## stmt\_history\_id - Statement history identifier

This numeric element shows the position in which the statement was run within the unit of work indicated by the sequence\_no element, relative to other statement history elements. The earliest statement run in the unit of work will have the lowest value. If the same statement is run twice in the same unit of work, two different occurrences of the statement will be shown with two different stmt\_history\_id values.

Table 1644. Event Monitoring Information

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_stmt_history    | -              |
| Deadlocks with Details History Values | event_data_value      | -              |

Table 1644. Event Monitoring Information (continued)

| Event Type                     | Logical Data Grouping | Monitor Switch |
|--------------------------------|-----------------------|----------------|
| Deadlocks with Details History | event_stmt_history    | -              |

**Usage** You can use this information to see the sequence of SQL statements that caused the deadlock.

## inact\_stmthist\_sz - Statement history list size

When a detailed deadlock event monitor with history is running, this element reports the number of bytes being used from the database monitor heap (MON\_HEAP\_SZ) to keep track of the statement history list entries.

Table 1645. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| application    | appl                  | -              |
| database       | db                    | -              |

**Usage** You can use this element when tuning the database monitor heap.

## stmt\_invocation\_id - Statement invocation identifier monitor element

An identifier that distinguishes one invocation of a routine from others at the same nesting level within a unit of work. It is unique within a unit of work for a specific nesting level.

Table 1646. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 1647. Event Monitoring Information

| Event Type   | Logical Data Grouping         | Monitor Switch |
|--|-------------------------------|----------------|
| Activities   | event_activitystmt            | -              |
| Locking  | -                             | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history            | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history            | -              |
| Unit of work                                       | Reported in the package list. | -              |

- 1** This option has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element to uniquely identify the invocation in which a particular SQL statement has been executed. You can also use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

---

## stmt\_isolation - Statement isolation

This element shows the isolation value in effect for the statement while it was being run.

Table 1648. Event Monitoring Information

| Event Type                               | Logical Data Grouping | Monitor Switch |
|--|-----------------------|----------------|
| Deadlocks with Details<br>History Values | event_stmt_history    | -              |
| Deadlocks with Details<br>History        | event_stmt_history    | -              |
| Activities                               | event_activitystmt    | -              |

The possible isolation level values are:

- SQLM\_ISOLATION\_LEVEL\_NONE 0 (no isolation level specified)
- SQLM\_ISOLATION\_LEVEL\_UR 1 (uncommitted read)
- SQLM\_ISOLATION\_LEVEL\_CS 2 (cursor stability)
- SQLM\_ISOLATION\_LEVEL\_RS 3 (read stability)
- SQLM\_ISOLATION\_LEVEL\_RR 4 (repeatable read)

**Usage** You can use this element in conjunction with other statement history entries to understand the cause of the deadlock and the execution behavior of a particular SQL statement.

---

## stmt\_last\_use\_time - Statement last use timestamp monitor element

This element shows the last time the statement entry was processed. For cursor operations, **stmt\_last\_use\_time** shows the time of the last action on the cursor where that action could be an open, fetch, or close. At application coordination nodes, this value reflects the application requests; at non-coordinator nodes, this value reflects when requests were received from the originating node.

Table 1649. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | event_stmt_history    | timestamp      |
| Deadlocks with Details<br>History <sup>1</sup>        | event_stmt_history    | timestamp      |
| Activities  | event_activitystmt    | timestamp      |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

---

### stmt\_lock\_timeout - Statement lock timeout monitor element

This element shows the lock timeout value in effect for the statement while it was being run.

Table 1650. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--|-----------------------|----------------|
| Locking  | -                     | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history    | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history    | -              |
| Activities   | event_activitystmt    | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element in conjunction with other statement history entries to understand the cause of the deadlock and the execution behavior of a particular SQL statement.

---

### stmt\_nest\_level - Statement nesting level monitor element

This element shows the level of nesting or recursion in effect when the statement was being run; each level of nesting corresponds to nested or recursive invocation of a stored procedure or user-defined function (UDF).

Table 1651. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 1652. Event Monitoring Information

| Event Type   | Logical Data Grouping         | Monitor Switch |
|--|-------------------------------|----------------|
| Locking  | -                             | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history            | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history            | -              |
| Activities   | event_activitystmt            | -              |
| Unit of work                                       | Reported in the package list. | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element, along with `stmt_invocation_id` monitor element, to uniquely identify the invocation in which a particular SQL statement has been executed. You can also use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

---

## stmt\_node\_number - Statement Node

Node where the statement was executed.

*Table 1653. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

**Usage** Used to correlate each statement with the node where it was executed.

---

## stmt\_operation/operation - Statement operation monitor element

The statement operation currently being processed or most recently processed (if none currently running).

*Table 1654. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| SNAPSHOT_STATEMENT table function   | ACTIVITY METRICS BASE            |
| SNAPSTMT administrative view and SNAP_GET_STMT table function - Retrieve statement snapshot information | ACTIVITY METRICS BASE            |

*Table 1655. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

*Table 1656. Event Monitoring Information*

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |
| Statements                          | event_stmt            | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element to determine the operation that is executing or recently finished.

It can be one of the following values:

For SQL operations:

- SELECT
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- OPEN
- FETCH
- CLOSE
- DESCRIBE
- STATIC COMMIT
- STATIC ROLLBACK
- FREE LOCATOR
- PREP\_COMMIT
- CALL
- PREP\_OPEN
- PREP\_EXEC
- COMPILE
- DROP PACKAGE

For non-SQL operations:

- RUN STATISTICS
- REORG
- REBIND
- REDISTRIBUTE
- GET TABLE AUTHORIZATION
- GET ADMINISTRATIVE AUTHORIZATION

**Note:** API users should refer to the `sqlmon.h` header file containing definitions of database system monitor constants.

---

## stmt\_pkgcache\_id - Statement package cache identifier monitor element

This element shows the internal package cache identifier (ID) for a dynamic SQL statement.

*Table 1657. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 1657. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1658. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

Table 1659. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch    |
|--|-----------------------|-------------------|
| Locking  | -                     | Always collected  |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history    | Always collected  |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history    | Always collected  |
| Activities   | event_activitystmt    | Always collected  |
| Package cache                                      | -                     | COLLECT BASE DATA |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

In a multi-partitioned environment, each partition has a unique statement ID for a cached statement. A given statement may not have the same ID across partitions.

In a global dynamic SQL snapshot, only the first statement ID is returned.

---

## stmt\_query\_id - Statement query identifier monitor element

This element shows the internal query identifier (ID) given to any SQL statement used as a cursor.

Table 1660. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--|-----------------------|----------------|
| Locking  | -                     | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history    | -              |
| Deadlocks with Details History <sup>1</sup>        | event_stmt_history    | -              |
| Activities   | event_activitystmt    | -              |

## Usage

You can use this element, along with the **stmt\_nest\_level** monitor element, to uniquely identify an invocation of a particular SQL statement. You can also use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## stmt\_sorts - Statement Sorts

The total number of times that a set of data was sorted in order to process the stmt\_operation.

Table 1661. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Statement      |
| Application    | stmt                  | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

**Usage** You can use this element to help identify the need for an index, since indexes can reduce the need for sorting of data. Using the related elements in the previously shown table you can identify the SQL statement for which this element is providing sort information, and then analyze this statement to determine index candidates by looking at columns that are being sorted (for example, columns used in ORDER BY and GROUP BY clauses and join columns). See **explain** in the *Administration Guide* for information on checking whether your indexes are used to optimize sort performance.

This count includes sorts of temporary tables that were generated internally by the database manager to execute the statement. The number of sorts is associated with the first FETCH operation of the SQL statement. This information is returned to you when the operation for the statement is the first FETCH. You should note that for blocked cursors several fetches may be performed when the cursor is opened. In these cases it can be difficult to use the snapshot monitor to obtain the number of sorts, since a snapshot would need to be taken while DB2 was internally issuing the first FETCH.

A more reliable way to determine the number of sorts performed when using a blocked cursor would be with an event monitor declared for statements. The total\_sorts counter, in the statement event for the CLOSE cursor, contains the total number of sorts that were performed while executing the statement for which the cursor was defined.

---

## stmt\_source\_id - Statement source identifier

This element shows the internal identifier (ID) given to the source of the SQL statement that was run.

Table 1662. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--|-----------------------|----------------|
| Locking  | -                     | -              |
| Deadlocks with Details History Values <sup>1</sup> | event_stmt_history    | -              |

Table 1662. Event Monitoring Information (continued)

| Event Type                                  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Deadlocks with Details History <sup>1</sup> | event_stmt_history    | -              |
| Activities                                  | event_activitystmt    | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element, along with **appl\_id** monitor element, to uniquely identify the origin of a request to run a particular SQL statement. You can also use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## stmt\_start - Statement Operation Start Timestamp

The date and time when the stmt\_operation started executing.

**Element identifier**  
stmt\_start

**Element type**  
timestamp

Table 1663. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | stmt                  | Statement, Timestamp |
| DCS Statement  | dcs_stmt              | Statement, Timestamp |

**Usage** You can use this element with stmt\_stop to calculate the elapsed statement operation execution time.

---

## stmt\_stop - Statement Operation Stop Timestamp

The date and time when the stmt\_operation stopped executing.

**Element identifier**  
stmt\_stop

**Element type**  
Timestamp

Table 1664. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | stmt                  | Statement, Timestamp |
| DCS Statement  | dcs_stmt              | Statement, Timestamp |

**Usage** You can use this element with stmt\_start to calculate the elapsed statement operation execution time.

---

## stmt\_sys\_cpu\_time - System CPU Time used by Statement

The total *system* CPU time (in seconds and microseconds) used by the currently executing statement.

**Element identifier**

stmt\_sys\_cpu\_time

**Element type**

time

*Table 1665. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | appl                  | Statement, Timestamp |
| Application    | stmt                  | Statement, Timestamp |

**Usage** This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

This counter includes time spent on both SQL and non-SQL statements, as well as any unfenced user defined functions (UDF) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

**Note:** If this information is not available for your operating system, this element will be set to 0.

---

## stmt\_text - SQL statement text monitor element

The text of the SQL statement.

*Table 1666. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE            |

*Table 1667. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| Dynamic SQL    | dynsql                | Basic          |
| DCS Statement  | dcs_stmt              | Statement      |

Table 1668. Event Monitoring Information

| Event Type                                  | Logical Data Grouping | Monitor Switch    |
|---|-----------------------|-------------------|
| Locking                                     | -                     | Always collected  |
| Deadlocks with Details <sup>1</sup>         | event_detailed_dlconn | Always collected  |
| Deadlocks with Details History <sup>1</sup> | event_stmt_history    | Always collected  |
| Statements                                  | event_stmt            | Always collected  |
| Activities                                  | event_activitystmt    | Always collected  |
| Package cache                               | -                     | COLLECT BASE DATA |
| Change history                              | ddlstmtexec           | Always collected  |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

For application snapshots, this statement text helps you identify what the application was executing when the snapshot was taken, or most recently processed if no statement was being processed right at the time the snapshot was taken.

The information returned by this element is taken from the SQL statement cache and it might not be available if the cache has overflowed. The only guaranteed way to capture the SQL text of a statement is to use an event monitor for statements.

For dynamic SQL statements, this element identifies the SQL text associated with a package.

For statement event monitors, this element is returned only for dynamic statements. If a statement event monitor record cannot fit into the size of the buffer specified by the BUFFERSIZE option of a statement event monitor, the value of the **stmt\_text** monitor may be truncated so that the record can fit.

For the EVENT\_STMT\_HISTORY event monitor, this element is returned only for dynamic statements. For remaining event monitors, **stmt\_text** is returned for dynamic and static statements only if it is available in the SQL statement cache.

For information about how to query the system catalog tables to obtain static SQL statement text that is not provided due to performance considerations, see the **section\_number** monitor element.

---

## stmt\_type - Statement type monitor element

The type of statement processed.

Table 1669. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 1670. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |
| Statements                          | event_stmt            | Always collected |
| Activities                          | event_activitystmt    | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element to determine the type of statement that is executing. It can be one of the following statements:

- A static SQL statement
- A dynamic SQL statement
- An operation other than an SQL statement; for example, a bind or pre-compile operation.

For the snapshot monitor, this element describes the statement that is currently being processed or was most recently processed.

**Note:** API users should refer to the `sqlmon.h` header file containing definitions of database system monitor constants.

---

## stmt\_type\_id - Statement type identifier monitor element

Statement type identifier.

Table 1671. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

Table 1672. Event Monitoring Information

| Event Type    | Logical Data Grouping | Monitor Switch    |
|---------------|-----------------------|-------------------|
| Package cache | -                     | COLLECT BASE DATA |

## Usage

The `stmt_type_id` monitor element has the following possible values:

- Statement not prepared
- DDL, (not Set Constraints)
- DDL, Set Constraints

- DML, Select
- DML, Insert/Update/Delete
- Authorization
- DML, Select (blockable)
- DML, Lock Table
- DML, Commit/Rollback
- Set environment
- DDL, Savepoint
- DDL, (declared user temp)
- Passthru support
- CALL
- Free locator
- DML, Select with IUD
- DML, Select with IUD (blockable)
- Top-level SET, no SQL
- Top-level SET, reads SQL
- DDL, (issues internal commit)
- Top-level SET, modifies SQL
- Unknown

---

## stmt\_unicode - Statement unicode flag monitor element

The SQL statement unicode flag. Possible values: Yes or No.

*Table 1673. Event Monitoring Information*

| Event Type | Logical Data Grouping       | Monitor Switch |
|------------|-----------------------------|----------------|
| Locking    | lock_participant_activities |                |

---

## stmt\_usr\_cpu\_time - User CPU Time used by Statement

The total *user* CPU time (in seconds and microseconds) used by the currently executing statement.

**Element identifier**

stmt\_usr\_cpu\_time

**Element type**

time

*Table 1674. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | appl                  | Statement, Timestamp |
| Application    | stmt                  | Statement, Timestamp |

**Usage** This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

This counter includes time spent on both SQL and non-SQL statements, as well as any unfenced user-defined functions (UDFs) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

**Note:** If this information is not available for your operating system, this element will be set to 0.

---

## stmt\_value\_data - Value data

This element contains a string representation of a data value to an SQL statement. LOB, LONG, and structured type parameters appear as empty strings. Date, time, and timestamp fields are recorded in ISO format.

*Table 1675. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

*Table 1676. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | stmt_value_data       | -              |
| Activities  | event_activityvals    | -              |

**1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

You can use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## stmt\_value\_index - Value index

This element represents the position of the input parameter marker or host variable used in the SQL statement.

*Table 1677. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

*Table 1678. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | stmt_value_data       | -              |

Table 1678. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activityvals    | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

You can use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## stmt\_value\_isnull - Value has null value monitor element

This element shows whether a data value that is associated with an SQL statement is the NULL value; whether an extended indicator has been used to specify the default value; or that this statement value is unassigned.

Possible values are:

- 0 or "no" if the value is not NULL
- 1 or "yes" if the value is NULL
- 2 or "default" if the extended indicator value of default (-5) was specified for this statement value
- 3 or "unassigned" if the extended indicator value of unassigned (-7) was specified for this statement value

Table 1679. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

Table 1680. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | stmt_value_isnull     | -              |
| Activities  | event_activityvals    | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

You can use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## stmt\_value\_isreopt - Variable used for statement reoptimization monitor element

This element shows whether the provided value was a value used during statement reoptimization. It returns a value of "True" if the statement was reoptimized (for example, due to the setting of the REOPT bind option) and if the value was used as input to the SQL compiler during this reoptimization.

*Table 1681. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

*Table 1682. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | event_data_value      | -              |
| Activities  | event_activityvals    | -              |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

### Usage

You can use this element in conjunction with the provided compilation environment to allow for full analysis of the SQL compiler's treatment of the SQL statement.

---

## stmt\_value\_type - Value type monitor element

This element contains a string representation of the type of a data value associated with an SQL statement.

*Table 1683. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS -<br>Get detailed metrics for package cache entries | ACTIVITY METRICS BASE            |

*Table 1684. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|---|-----------------------|----------------|
| Locking   | -                     | -              |
| Deadlocks with Details<br>History Values <sup>1</sup> | stmt_value_type       | -              |
| Activities  | event_activityvals    | -              |

- 1** This event monitor has been deprecated. Its use is no longer recommended

and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

You can use this element in conjunction with other statement history entries to understand the cause of the deadlock.

---

## sto\_path\_free\_sz - Automatic storage path free space monitor element

This element shows the amount of free space (in bytes) available on a file system pointed to by a storage path. If multiple storage paths point to the same file system, the free size is not divided among them.

*Table 1685. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups | ACTIVITY METRICS BASE            |

*Table 1686. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Buffer Pool    |

## Usage

You can use this element together with the following elements to gather per-node data on space utilization for the database:

- **db\_storage\_path**
- **fs\_used\_size**
- **fs\_total\_size**
- **fs\_id**

---

## stop\_time - Event Stop Time

The date and time when the statement stopped executing.

*Table 1687. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | Timestamp      |

**Usage** You can use this element with *start\_time* to calculate the elapsed statement execution time.

For a FETCH statement event, this is the time of the last successful fetch.

**Note:** When the Timestamp switch is OFF, this element reports "0".

---

## storage\_group\_id - Storage group identifier

An integer that uniquely represents a storage group used by the current database.

*Table 1688. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - get table space metrics                              | ACTIVITY METRICS BASE            |

### Usage notes

- If using the ADMIN\_GET\_STORAGE\_PATHS table function, the storage group identifier indicates the storage group to which a storage path is defined.
- If using the MON\_GET\_TABLESPACES table function, the storage group identifier indicates which storage group the table space is defined in.

---

## storage\_group\_name - Storage group name

Name of a storage group.

*Table 1689. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_STORAGE_PATHS table function - get storage path information for storage groups | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - get table space metrics                              | ACTIVITY METRICS BASE            |

### Usage notes

- If using the ADMIN\_GET\_STORAGE\_PATHS table function, this monitor element indicates the storage group to which a storage path is defined.
- If using the MON\_GET\_TABLESPACES table function, this monitor element indicates which storage group the table space is defined in.

---

## stored\_proc\_time - Stored Procedure Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to stored procedure statements from all applications or a single application running on this federated server instance from the start of the federated server instance or the last reset of the database monitor counters.

*Table 1690. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

The response time is measured as the difference between the time the federated server submits a stored procedure to the data source, and the time it takes the data source to respond, indicating that the stored procedure has been processed.

**Usage** Use this element to determine how much actual time is spent at this data source processing stored procedures.

---

## stored\_procs - Stored Procedures

This element contains a count of the total number of stored procedures from the start of the federated server instance, or the last reset of the database monitor counters, that the federated server has called at this data source on behalf of any application.

*Table 1691. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine how many stored procedure calls were made locally at the federated database or by an application against the federated database.

---

## swap\_pages\_in - Pages swapped in from disk monitor element

The number of pages swapped in from disk since system startup. Reported for AIX and Linux systems only.

*Table 1692. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## swap\_pages\_out - Pages swapped out to disk monitor element

The number of pages swapped out to disk since since system startup. Reported for AIX and Linux systems only.

*Table 1693. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## swap\_page\_size - Swap page size monitor element

The page size used for swap space, in bytes. Reported for AIX and Linux systems only.

*Table 1694. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table function - Return system information | ACTIVITY METRICS BASE            |

---

## sync\_runstats – Total number of synchronous RUNSTATS activities monitor element

The total number of synchronous RUNSTATS activities triggered by real-time statistics gathering for all the applications in the database. This value includes both successful and unsuccessful synchronous RUNSTATS commands. Values reported by all the database partitions are aggregated together.

*Table 1695. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

*Table 1696. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

*Table 1697. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

### Usage

Use this monitor element to determine how many synchronous RUNSTATS activities have been triggered by real-time statistics gathering in the database. This value changes frequently. In order to get a better view of the system usage, take a snapshot at specific intervals over an extended period of time. When used in conjunction with **sync\_runstats\_time**, this element can help you evaluate the performance impact of synchronous RUNSTATS activities triggered by real-time statistics gathering.

---

## sync\_runstats\_time – Total time spent on synchronous RUNSTATS activities monitor element

The total time spent on synchronous RUNSTATS activities triggered by real-time statistics gathering, in milliseconds. The synchronous RUNSTATS activities occur during query compilation. At the database level, this monitor element represents the total time spent on synchronous RUNSTATS activities for all the applications running on the database, triggered by real-time statistics gathering. At the statement level, it represents the time spent on the latest synchronous RUNSTATS activities for a particular statement, triggered by real-time statistics gathering. Values reported by all the database partitions are aggregated together.

*Table 1698. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group                            |                                  |
| SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group |                                  |

*Table 1699. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this element can be reset.

*Table 1700. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Statement  | event_stmt            | Always collected |

### Usage

Use this element along with **sync\_runstats** to evaluate the performance impact of synchronous RUNSTATS activities triggered by real-time statistics gathering, at the database level,

For dynamic SQL snapshot monitor, use this element along with **total\_exec\_time** and **num\_executions** to evaluate the impact of synchronous RUNSTATS on query performance.

For the statement event monitor, use this element along with **stmt\_start** and **stmt\_stop** for further evaluation of the impact of real-time statistics gathering.

---

## system\_auth\_id - System authorization identifier monitor element

The system authorization id for the connection.

Table 1701. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                          | ACTIVITY METRICS BASE            |

Table 1702. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Element Collection Level |
|----------------|-----------------------|----------------------------------|
| Change history | changesummary         | Always collected                 |

---

## system\_cpu\_time - System CPU time monitor element

The total *system* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

When either the statement monitor switch or the timestamp switch is not turned on, this element is not collected. In that case, the monitor element displays -1 instead.

Table 1703. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Connection   | event_conn            | Always collected |
| Transactions | event_xact            | Always collected |
| Statements   | event_stmt            | Always collected |
| Activities   | event_activity        | Always collected |

### Usage

This element, along with the other related CPU-time elements, can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

**Note:** If this information is not available for your operating system, this element will be set to 0.

**Note:** Due to the differences in granularity with which the DB2 system collects statistics, the value of the **total\_exec\_time** monitor element might not equal the sum of values of **system\_cpu\_time** and **user\_cpu\_time** monitor elements. In this

case, the sum of **system\_cpu\_time** and **user\_cpu\_time** monitor elements more accurately reflects the actual total execution time.

---

## tab\_file\_id - Table file ID monitor element

A file ID (FID) for the table.

*Table 1704. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLE_METRICS table function<br>- Get table metrics | ACTIVITY METRICS BASE                        |

### Usage

---

## tab\_type - Table type monitor element

This interface returns a text identifier based on defines in sqlmon.h, and is one of USER\_TABLE, TEMP\_TABLE, or CATALOG\_TABLE.

*Table 1705. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE_METRICS table function<br>- Get table metrics | ACTIVITY METRICS BASE            |

### Usage

---

## table\_file\_id - Table file ID monitor element

The file ID (FID) for the table.

*Table 1706. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables | ACTIVITY METRICS BASE            |
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting                   | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics   | ACTIVITY METRICS BASE            |

*Table 1707. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Table          | table                 | Basic          |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Lock           |

Table 1708. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Deadlocks  | lock                  | Always collected |

## Usage

For snapshot monitoring, this element is provided for information purposes only. It is returned for compatibility with previous versions of the database system monitor, and it may not uniquely identify the table. Use **table\_name** and **table\_schema** monitor elements to identify the table.

In MON\_GET\_LOCKS and MON\_GET\_APPL\_LOCKWAIT table functions, this element represents the file ID (FID) for the table that the lock references.

---

## table\_name - Table name monitor element

The name of the table.

Table 1709. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO table function - returns compressed index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function - returns index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries                                   | ACTIVITY METRICS BASE            |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information                | ACTIVITY METRICS BASE            |
| ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS table function - Retrieve column information for temporary tables | ACTIVITY METRICS BASE            |
| ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables          | ACTIVITY METRICS BASE            |
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details  | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics  | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics  | ACTIVITY METRICS BASE            |

Table 1709. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list | ACTIVITY METRICS BASE            |

Table 1710. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 1711. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Tables                              | event_table           | Always collected |
| Deadlocks <sup>1</sup>              | lock                  | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

- <sup>1</sup> This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Along with **table\_schema**, this element can help you determine the source of contention for resources.

At the application-level, application-lock level, and deadlock-monitoring-level, this is the table that the application is waiting to lock, because it is currently locked by another application. For snapshot monitoring, this item is only valid when the “lock” monitor group information is set to ON, and when **lock\_object\_type** indicates that the application is waiting to obtain a table lock.

For snapshot monitoring at the object-lock level, this item is returned for table-level and row-level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this is the table for which information has been collected. For temporary tables, the format for **table\_name** is “TEMP (*n*, *m*)”, where:

- *n* is the table space ID
- *m* is the **table\_file\_id** element

---

## table\_scans - Table scans monitor element

The number of scans on this table.

Table 1712. Table Function Monitoring Information

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

---

### Usage

---

## table\_schema - Table schema name monitor element

The schema of the table.

Table 1713. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO table function - returns compressed index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function - returns index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries                                   | ACTIVITY METRICS BASE            |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information                | ACTIVITY METRICS BASE            |
| ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS table function - Retrieve column information for temporary tables | ACTIVITY METRICS BASE            |
| ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables          | ACTIVITY METRICS BASE            |
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details  | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics  | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list   | ACTIVITY METRICS BASE            |

---

Table 1714. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 1715. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Tables                              | event_table           | Always collected |
| Deadlocks <sup>1</sup>              | lock                  | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Along with **table\_name**, this element can help you determine the source of contention for resources.

For application-level, application-lock-level, deadlock-monitoring-level, this is the schema of the table that the application is waiting to lock, because it is currently locked by another application. This element is only set if **lock\_object\_type** indicates that the application is waiting to obtain a table lock. For snapshot monitoring at the application-level and application-lock levels, this item is only valid when the “lock” monitor group information is set to ON.

For snapshot monitoring at the object-lock level, this item is returned for table and row level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this element identifies the schema of the table for which information has been collected. For temporary tables, the format for **table\_schema** is “<agent\_id><auth\_id>”, where:

- *agent\_id* is the Application Handle of the application creating the temporary table
- *auth\_id* is the authorization ID used by the application to connect to the database

---

## table\_type - Table type monitor element

The type of table for which information is returned.

*Table 1716. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

*Table 1717. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 1718. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

### Usage

Use this element to help identify the table for which information is returned. If the table is a user table or a system catalog table, you can use **table\_name** and **table\_schema** to identify the table.

The type of table may be one of the following values. The possible values are text strings based on defines in the `sqlmon.h` file.

#### **USER\_TABLE**

User table.

#### **TEMP\_TABLE**

Temporary table. Information regarding temporary tables is returned, even though the tables are not kept in the database after being used. You may still find information about this type of table useful.

#### **CATALOG\_TABLE**

System catalog table.

---

## tablespace\_auto\_resize\_enabled - Table space automatic resizing enabled monitor element

This element describes whether automatic resizing is enabled for the table space. A value of 1 means "Yes"; a value of 0 means "No".

*Table 1719. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1720. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

## Usage

This element is only applicable to DMS table spaces and non-temporary automatic storage table spaces. If this element is set to 1, then automatic resizing is enabled. See the following monitor elements for information about the rate of increase and the maximum size for the table space.

- **tablespace\_max\_size**
- **tablespace\_increase\_size**
- **tablespace\_increase\_size\_percent**

---

## tablespace\_content\_type - Table space content type monitor element

The type of content in a table space.

*Table 1721. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1722. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

## Usage

The type of content in the table space (defined in sqlmon.h) can be one of the following values:

- All types of permanent data.
  - Regular table space: `SQLM_TABLESPACE_CONTENT_ANY`
  - Large table space: `SQLM_TABLESPACE_CONTENT_LARGE`
- System temporary data: `SQLM_TABLESPACE_CONTENT_SYSTEMP`
- User temporary data: `SQLM_TABLESPACE_CONTENT_USRTEMP`

---

## tablespace\_cur\_pool\_id - Buffer pool currently being used monitor element

The buffer pool identifier for a buffer pool that a table space is currently using.

*Table 1723. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE                        |

*Table 1724. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

**Usage** Each buffer pool is identified by a unique integer. The value of this element matches a value from column BUFFERPOOLID of view SYSCAT.BUFFERPOOLS.

---

## tablespace\_current\_size - Current table space size

This element shows the current size of the table space in bytes.

*Table 1725. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** For DMS and automatic storage table spaces, this element represents the total size of all table space containers in bytes. This value is equal to the total pages for the table space (tablespace\_total\_pages) multiplied by the table space's page size (tablespace\_page\_size). This element is not applicable for SMS table spaces, or for temporary automatic storage table spaces.

On table space creation for an automatic storage table space, the current size might not match the initial size. The value of current size will be within page size multiplied by extent size multiplied by the number of storage paths of the initial size on creation (usually greater, but sometimes smaller). It will always be less than or equal to tablespace\_max\_size (if set). This is because containers can only grow by full extents, and must be grown as a set.

---

## tablespace\_extent\_size - Table space extent size monitor element

The extent size used by a table space.

*Table 1726. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

*Table 1727. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

---

## tablespace\_free\_pages - Free pages in table space monitor element

The total number of pages that are currently free in a table space.

*Table 1728. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

Table 1729. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

## Usage

This is applicable only to a DMS table space.

---

## tablespace\_id - Table space identification monitor element

An integer that uniquely represents a table space used by the current database.

Table 1730. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables | ACTIVITY METRICS BASE            |
| MON_GET_APPL_LOCKWAIT table function - get information about locks for which an application is waiting                   | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics   | ACTIVITY METRICS BASE            |
| MON_GET_EXTENT_MOVEMENT_STATUS - Get extent movement progress status metrics   | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all locks in the currently connected database  | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space                                       | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics   | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics  | ACTIVITY METRICS BASE            |

Table 1731. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |
| Table          | table                 | Basic          |

Table 1732. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

## Usage

The value of this element matches a value from column TBSPACEID of view SYSCAT.TABLESPACES.

---

## tablespace\_increase\_size - Increase size in bytes

This element shows the size that an auto-resize table space will increase by in bytes when the table space becomes full and more space is required.

Table 1733. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This represents the amount of space that will be added to a table space that can be automatically resized when it becomes full, more space is being requested, and the maximum table space size has not been reached. If the value of this element is -1 (or "AUTOMATIC" in the snapshot output), then DB2 automatically determines the value when space needs to be added. This element is only applicable to table spaces that are enabled to be automatically resized.

---

## tablespace\_increase\_size\_percent - Increase size by percent monitor element

This element shows the amount by which an auto-resize table space will increase when the table space becomes full and more space is required. The actual number of bytes is determined at the time the table space is resized based on the size of the table space at that time.

Table 1734. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This represents the amount of space that will be added to a table space that can be automatically resized when it becomes full, more space is being requested, and the maximum table space size has not been reached. The growth rate is based on a percentage of the current table space size (tablespace\_current\_size) at the time the table space is resized. This element is only applicable to table spaces that are enabled to be automatically resized.

---

## tablespace\_initial\_size - Initial table space size

The initial size of the automatic storage table space in bytes.

Table 1735. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** For non-temporary automatic storage table spaces, this monitor element represents the initial size in bytes for the table space when it was created.

---

## tablespace\_last\_resize\_failed - Last resize attempt failed

This element describes whether or not the last attempt to automatically increase the size of the table space failed. A value of 1 means yes, 0 means no.

Table 1736. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** For an automatic storage table space, this element may show that there is no space left on any of the database's storage paths. For a non-automatic storage table space, a failure means that one of the containers could not be extended because its filesystem was full. Another reason for failure is that the maximum size of the table space has been reached. This element is only applicable to table spaces that are enabled to be automatically resized.

---

## tablespace\_last\_resize\_time - Time of last successful resize

This element shows a timestamp representing the last time that the size of the table space was successfully increased.

Table 1737. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** For table spaces that can be automatically resized, this element represents the last time that space was automatically added to the table space when it became full, more space was being requested, and the maximum table space size had not been reached. This element is only applicable to table spaces that are enabled to be automatically resized.

---

## tablespace\_max\_size - Maximum table space size

This element shows the maximum size in bytes to which the table space can automatically resize or increase.

Table 1738. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

Table 1739. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This represents the maximum size in bytes to which a table space that can be automatically resized can automatically increase. If this value is equal to the tablespace\_current\_size element, then there is no room for the table space to grow. If the value of this element is -1, then the maximum size is considered to be "unlimited" and the table space can automatically resize until the file systems are full or the architectural size limit of the table

space is reached. (This limit is described in the SQL Limits appendix of the *SQL Reference*). This element is only applicable to table spaces that are enabled for automatic resizing.

---

## tablespace\_min\_recovery\_time - Minimum recovery time for rollforward monitor element

A timestamp showing the earliest point in time to which a table space can be rolled forward. The timestamp reflects local time.

*Table 1740. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1741. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

Displayed only if non zero.

---

## tablespace\_name - Table space name monitor element

The name of a table space.

*Table 1742. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics                   | ACTIVITY METRICS BASE            |
| MON_GET_EXTENT_MOVEMENT_STATUS - Get extent movement progress status metrics           | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS - Get rebalance progress for a table space                    | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics                            | ACTIVITY METRICS BASE            |

*Table 1743. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |
| Lock           | appl_lock_list        | Basic          |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 1744. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch |
|-------------------------------------|-----------------------|----------------|
| Locking                             | -                     | -              |
| Deadlocks <sup>1</sup>              | lock                  | -              |
| Deadlocks <sup>1</sup>              | event_dlconn          | -              |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | -              |
| Table Space                         | tablespace_list       | -              |

- 1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

This element can help you determine the source of contention for resources.

It is equivalent to the TBSpace column in the database catalog table SYSCAT.TABLESPACES. At the application level, application-lock level, and deadlock monitoring level, this is the name of the table space that the application is waiting to lock. Another application currently holds a lock on this table space.

At the lock level, this is the name of the table space against which the application currently holds a lock.

At the table space level (when the buffer pool monitor group is ON), this is the name of the table space for which information is returned.

This element will not be returned for a table lock held on a partitioned table.

---

## tablespace\_next\_pool\_id - Buffer pool that will be used at next startup monitor element

The buffer pool identifier for a buffer pool that a table space will use at the next database startup.

Table 1745. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

Table 1746. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

**Usage** Each buffer pool is identified by a unique integer. The value of this element matches a value from column BUFFERPOOLID of view SYSCAT.BUFFERPOOLS

---

## tablespace\_num\_containers - Number of Containers in Table Space

Total number of containers in the table space.

*Table 1747. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

---

---

## tablespace\_num\_quiescers - Number of Quiescers

The number of users quiescing the table space (can be in the range of 0 to 5).

*Table 1748. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

---

**Usage** This value represents the number of agents that have quiesced the table space (either in "SHARE", "UPDATE", or "EXCLUSIVE" mode). For each quiescer, the following information is returned in a tablespace\_quiescer logical data group:

- User authorization ID of the quiescer
- Agent ID of the quiescer
- Table space ID of the object that was quiesced that resulted in this table space being quiesced
- Object ID of the object that was quiesced that resulted in this table space being quiesced
- Quiesce state

---

## tablespace\_num\_ranges - Number of Ranges in the Table Space Map

The number of ranges (entries) in the table space map. This can be in the range of 1 to 100's (but is usually less than a dozen). The table space map only exists for DMS table spaces.

*Table 1749. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

---

---

## tablespace\_page\_size - Table space page size monitor element

Page size used by a table space in bytes.

*Table 1750. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

---

Table 1751. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

---

## tablespace\_page\_top - Table space high watermark monitor element

The page in a table space that is holding the high watermark.

Table 1752. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

Table 1753. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

For DMS, this element represents the page number of the first free extent following the last allocated extent of a table space. Note that this is not really a "high watermark", but rather a "current watermark", since the value can decrease. For SMS, this is not applicable.

---

## tablespace\_paths\_dropped - Table space using dropped path monitor element

Indicates that the table space is using a storage path that has been dropped.

Table 1754. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

Table 1755. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

For table spaces using automatic storage, use this monitor element to determine whether any of the table space containers reside on a storage path that has been dropped. Before storage paths are physically dropped from the database, all table spaces must stop using them. To stop using a dropped storage path, either drop the table space or rebalance the table space using the REBALANCE clause of the ALTER TABLESPACE statement.

---

## tablespace\_pending\_free\_pages - Pending free pages in table space monitor element

The number of pages in a table space which would become free if all pending transactions are committed or rolled back and new space is requested for an object.

Table 1756. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

Table 1757. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

This is applicable only to a DMS table space.

---

## tablespace\_prefetch\_size - Table space prefetch size monitor element

The maximum number of pages the prefetcher gets from the disk at a time.

Table 1758. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

Table 1759. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

- For table function monitoring, this element always reports the actual value for the table space prefetch size.
- For snapshot monitoring, if automatic prefetch size is enabled, this element reports the value "-1" in the *tablespace* Logical Data Grouping, and the actual value is reported in the *tablespace\_nodeinfo* Logical Data Grouping.
- For snapshot monitoring, if automatic prefetch size is not enabled, this element reports the actual value in the *tablespace* Logical Data Grouping, and the element does not appear in the *tablespace\_nodeinfo* Logical Data Grouping.

---

## tablespace\_rebalancer\_extents\_processed - Number of extents the rebalancer has processed

The number of extents that the rebalancer has already moved since the rebalancer has been started or restarted (whichever is most recent).

*Table 1760. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |

*Table 1761. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This can be used as an indicator of the completion level of the rebalancer. You can monitor the progress of rebalancing by keeping track of the change in this element over time. You can use `tablespace_state` and `rebalance_mode` to check if the rebalancing is completed. This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_extents\_remaining - Total number of extents to be processed by the rebalancer

The number of extents to be moved. This value is calculated at either the rebalancer start time or restart time (whichever is most recent).

*Table 1762. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |

*Table 1763. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This element can be used as an indicator of the completion level of the rebalancer. You can monitor the progress of rebalancing by keeping track of the change in this element over time. You can use `tablespace_state` to check if rebalancing has completed. This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_last\_extent\_moved - Last extent moved by the rebalancer

The last extent moved by the rebalancer.

Table 1764. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |

Table 1765. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This can be used as an indicator of the completion level of the rebalancer. You can monitor the progress of rebalancing by keeping track of the change in this element over time. You can use `tablespace_state` and `rebalance_mode` to check if the rebalancing is completed. This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_mode - Rebalancer mode monitor element

Indicates whether the current rebalance process is removing space from a table space or adding space to a table space.

Table 1766. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics                        | ACTIVITY METRICS BASE            |

Table 1767. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

A *forward rebalance* occurs when new containers are added or existing containers have their size increased. In a forward rebalancing operation, data movement starts with the first extent in the table space and ends with the high watermark extent.

A *reverse rebalance* occurs when containers are removed or reduced in size and data needs to move out of the space being freed. In a reverse rebalancing operation, data movement starts at the high watermark extent and moves in reverse order through the table space, ending with the first extent in the table space.

A *two-pass rebalance* is a forward rebalance followed by a reverse rebalance. A two-pass rebalance might occur when containers are being both added and dropped as part of the rebalance operation.

For DMS non-automatic storage table spaces, this monitor element indicates the type of rebalance that is occurring for the table space. Only a single forward rebalance or a single reverse rebalance can occur for DMS non-automatic table space.

For automatic storage table spaces, this monitor element indicates what the current rebalance process is doing to the table space. In general, only a single forward rebalance or a single reverse rebalance is necessary when a rebalance is initiated. However, there are cases when a two-pass rebalance is necessary for automatic storage table spaces.

The possible **tablespace\_rebalancer\_mode** values are defined in the `sqlmon.h` file. The following values are returned in snapshot monitoring:

**SQLM\_TABLESPACE\_NO\_REBAL**

No rebalancing is taking place.

**SQLM\_TABLESPACE\_FWD\_REBAL**

Forward rebalance is taking place.

**SQLM\_TABLESPACE\_REV\_REBAL**

Reverse rebalance is taking place.

**SQLM\_TABLESPACE\_FWD\_REBAL\_OF\_2PASS**

The forward rebalance phase of two pass rebalance is taking place.

**SQLM\_TABLESPACE\_REV\_REBAL\_OF\_2PASS**

The reverse rebalance phase of two pass rebalance is taking place.

If using either the `MON_GET_TABLESPACE` or the `MON_GET_REBALANCE_STATUS` table function, the following `rebalancer_mode` values are returned:

- NO\_REBAL
- FWD\_REBAL
- REV\_REBAL
- FWD\_REBAL\_OF\_2PASS
- REV\_REBAL\_OF\_2PASS

---

## tablespace\_rebalancer\_priority - Current rebalancer priority

The priority at which the rebalancer is running in the database.

*Table 1768. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |

*Table 1769. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_restart\_time - Rebalancer restart time

A timestamp representing when a rebalancer was restarted after being paused or suspended.

*Table 1770. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space | ACTIVITY METRICS BASE            |

*Table 1771. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This can be used as an indicator of the completion level of the rebalancer. It will note when the rebalancer was restarted, and will allow for the derivation of the speed of the rebalancer and the estimated time until completion. This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_source\_storage\_group\_id - Rebalancer source storage group identifier

The source storage group identifier if the rebalancer is moving a table space from one storage group to another. Otherwise, it is -1.

*Table 1772. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - Get rebalance progress for a table space | ACTIVITY METRICS BASE            |

---

## tablespace\_rebalancer\_source\_storage\_group\_name - Rebalancer source storage group name

The source storage group name if the rebalancer is moving a table space from one storage group to another. Otherwise, it is NULL.

*Table 1773. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - Get rebalance progress for a table space | ACTIVITY METRICS BASE            |

---

## tablespace\_rebalancer\_start\_time - Rebalancer start time

A timestamp representing when a rebalancer was initially started.

*Table 1774. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table<br>function - get rebalance progress for a table<br>space | ACTIVITY METRICS BASE            |

*Table 1775. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This will be used to note the time at which a rebalancer was initially started. This can be used to derive metrics as to the speed at which the rebalancer is operating, and the estimated time of completion of the rebalance. This is only applicable to a DMS table space.

---

## tablespace\_rebalancer\_status - Rebalancer status monitor element

Indicates the current status of the rebalance operation.

*Table 1776. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table<br>function - get rebalance progress for a table<br>space | ACTIVITY METRICS BASE            |

### Usage note

Current status of the rebalance operation is one of:

- ACTIVE - Rebalance operation is active.
- SUSPENDED - Rebalance operation has been explicitly suspended by a user using the ALTER TABLESPACE statement.
- PAUSED - Rebalance operation has been implicitly paused due to an online backup. The rebalance will proceed when the backup is complete.

If the rebalance operation has been explicitly suspended and implicitly paused, the status is reported as SUSPENDED.

---

## tablespace\_rebalancer\_target\_storage\_group\_id - Rebalancer target storage group identifier

The target storage group identifier if the rebalancer is moving a table space from one storage group to another. Otherwise, it is -1.

*Table 1777. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table<br>function - Get rebalance progress for a table<br>space | ACTIVITY METRICS BASE            |

This monitor element is connected to the `target_storage_group_name` monitor element, as well as `source_storage_group_id` monitor element and `source_storage_group_name` monitor element. You use these elements together to understand if a rebalance operation is moving a tablespace from one storage group to another, and to understand which storage group the tablespace is moving from (the source) and where it is moving to (the target).

---

## tablespace\_rebalancer\_target\_storage\_group\_name - Rebalancer target storage group name

The target storage group name if the rebalancer is moving a table space from one storage group to another. Otherwise, it is NULL.

*Table 1778. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_REBALANCE_STATUS table function - Get rebalance progress for a table space | ACTIVITY METRICS BASE            |

---

## tablespace\_state - Table space state monitor element

This element describes the current state of a table space.

*Table 1779. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1780. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

In administrative views and table functions, this monitor element returns a text identifier based on defines in `sqlutil.h`, and is combination of the following values separated by a '+' sign:

- BACKUP\_IN\_PROGRESS
- BACKUP\_PENDING
- DELETE\_PENDING
- DISABLE\_PENDING
- DROP\_PENDING
- LOAD\_IN\_PROGRESS
- LOAD\_PENDING
- MOVE\_IN\_PROGRESS
- NORMAL
- OFFLINE
- PSTAT\_CREATION

- PSTAT\_DELETION
- QUIESCED\_EXCLUSIVE
- QUIESCED\_SHARE
- QUIESCED\_UPDATE
- REBAL\_IN\_PROGRESS
- REDIST\_IN\_PROGRESS
- REORG\_IN\_PROGRESS
- RESTORE\_IN\_PROGRESS
- RESTORE\_PENDING
- ROLLFORWARD\_IN\_PROGRESS
- ROLLFORWARD\_PENDING
- STORDEF\_ALLOWED
- STORDEF\_CHANGED
- STORDEF\_FINAL\_VERSION
- STORDEF\_PENDING
- SUSPEND\_WRITE

This element contains a hexadecimal value indicating the current table space state. The externally visible state of a table space is composed of the hexadecimal sum of certain state values. For example, if the state is "quiesced: EXCLUSIVE" and "Load pending", the value is 0x0004 + 0x0008, which is 0x000c. Use the **db2tbst** command to obtain the table space state associated with a given hexadecimal value.

*Table 1781. Bit definitions listed in sqlutil.h*

| Hexadecimal Value | Decimal Value | State  |
|-------------------|---------------|--|
| 0x0               | 0             | Normal (see the definition SQLB_NORMAL in sqlutil.h) |
| 0x1               | 1             | Quiesced: SHARE                                      |
| 0x2               | 2             | Quiesced: UPDATE                                     |
| 0x4               | 4             | Quiesced: EXCLUSIVE                                  |
| 0x8               | 8             | Load pending   |
| 0x10              | 16            | Delete pending                                       |
| 0x20              | 32            | Backup pending                                       |
| 0x40              | 64            | Roll forward in progress                             |
| 0x80              | 128           | Roll forward pending                                 |
| 0x100             | 256           | Restore pending                                      |
| 0x100             | 256           | Recovery pending (not used)                          |
| 0x200             | 512           | Disable pending                                      |
| 0x400             | 1024          | Reorg in progress                                    |
| 0x800             | 2048          | Backup in progress                                   |
| 0x1000            | 4096          | Storage must be defined                              |
| 0x2000            | 8192          | Restore in progress                                  |
| 0x4000            | 16384         | Offline and not accessible                           |
| 0x8000            | 32768         | Drop pending   |
| 0x10000           | 65536         | No write is allowed                                  |

Table 1781. Bit definitions listed in sqlutil.h (continued)

| Hexadecimal Value | Decimal Value | State   |
|-------------------|---------------|---|
| 0x20000           | 131072        | Load in progress                                  |
| 0x40000           | 262144        | Redistribute in progress                          |
| 0x80000           | 524288        | Move in progress                                  |
| 0x2000000         | 33554432      | Storage may be defined                            |
| 0x4000000         | 67108864      | Storage Definition is in 'final' state            |
| 0x8000000         | 134217728     | Storage Definition was changed before rollforward |
| 0x10000000        | 268435456     | DMS rebalancer is active                          |
| 0x20000000        | 536870912     | TBS deletion in progress                          |
| 0x40000000        | 1073741824    | TBS creation in progress                          |

**Note:** DB2 LOAD does not set the table space state to Load pending or Delete pending.

---

## tablespace\_state\_change\_object\_id - State Change Object Identification

The object that caused the table space state to be set to "Load pending" or "Delete pending".

**Element identifier**

tablespace\_state\_change\_object\_id

**Element type**

information

Table 1782. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This element is meaningful only if the table space state is "Load pending" or "Delete pending". If nonzero, the value of this element matches a value from column TABLEID of view SYSCAT.TABLES.

**Note:** DB2 LOAD does not set the table space state to Load pending or Delete pending.

---

## tablespace\_state\_change\_ts\_id - State Change Table Space Identification

If the table space state is "Load pending" or "Delete pending", this shows the table space ID of the object that caused the table space state to be set.

**Element identifier**

tablespace\_state\_change\_ts\_id

**Element type**

information

Table 1783. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** This element is meaningful only if the table space state is "Load pending" or "Delete pending". If nonzero, the value of this element matches a value from column TABLESPACEID of view SYSCAT.TABLES.

**Note:** DB2 LOAD does not set the table space state to Load pending or Delete pending.

---

## tablespace\_total\_pages - Total pages in table space monitor element

Total number of pages in a table space.

Table 1784. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

Table 1785. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|--|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<br><br>Buffer Pool (SMS table spaces) |

### Usage

Total operating system space occupied by a table space. For DMS, this is the sum of the container sizes. For SMS, this is the sum of all file space used for the tables stored in this table space (and is only collected if the buffer pool switch is on).

---

## tablespace\_type - Table space type monitor element

The type of a table space.

Table 1786. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

Table 1787. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

### Usage

This element shows whether this table space is a database managed table space (DMS), or system managed table space (SMS).

The values for `tablespace_type` (defined in `sqlmon.h`) are as follows:

- For DMS: `SQLM_TABLESPACE_TYP_DMS`
- For SMS: `SQLM_TABLESPACE_TYP_SMS`

---

## tablespace\_usable\_pages - Usable pages in table space monitor element

The total number of pages in a table space minus overhead pages.

*Table 1788. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1789. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|--|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<br><br>Buffer Pool (SMS table spaces) |

### Usage

This element is applicable to DMS table spaces only. For SMS table spaces, this element will have the same value as the **tablespace\_total\_pages** monitor element.

During a table space rebalance, the number of usable pages will include pages for the newly added container, but these new pages may not be reflected in the number of free pages until the rebalance is complete. When a table space rebalance is not taking place, the number of used pages plus the number of free pages, plus the number of pending free pages will equal the number of usable pages.

---

## tablespace\_used\_pages - Used pages in table space monitor element

The total number of pages that are currently used (not free) in a table space.

*Table 1790. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE            |

*Table 1791. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|--|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<br><br>Buffer Pool (SMS table spaces) |

## Usage

This is the total number of pages in use for a DMS table space. For an SMS table space it is equal to the value of **tablespace\_total\_pages** monitor element.

---

## tablespace\_using\_auto\_storage - Table space enabled for automatic storage monitor element

This element describes whether the table space was created as an automatic storage table space. A value of 1 means "Yes"; a value of 0 means "No".

*Table 1792. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE                        |

*Table 1793. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

## Usage

You can use this element to determine whether the given table space was created using automatic storage (that is, created with the **MANAGED BY AUTOMATIC STORAGE** clause), rather than with containers that are explicitly provided. The table space can have containers that exist on some or all of the storage paths associated with the database.

---

## target\_cf\_gbp\_size - Target cluster caching facility group buffer pool size monitor element

During a dynamic resize, this monitor element shows the group buffer pool memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 1794. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## target\_cf\_lock\_size - Target cluster caching facility lock size monitor element

During a dynamic resize, this monitor element shows the global lock memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 1795. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## target\_cf\_sca\_size - Target cluster caching facility shared communications area size monitor element

During a dynamic resize, this monitor element shows the shared communications area memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 1796. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## tbsp\_datatag - Table space data tag

This element identifies the effective data tag value for a table space. The effective data tag is the data tag value that was specified explicitly for the table space or inherited from the table space storage group. Valid user-specified range is 1 through 0; 0 indicates no data tag specified.

*Table 1797. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - get table space metrics | ACTIVITY METRICS BASE            |

### Usage note

A data tag is used to identify and group data that can be referenced within WLM configurations. The WLM configurations determine the effect of the tagging which may affect the processing priority of user work.

---

## **tbsp\_last\_consec\_page - Last consecutive object table page monitor element**

Object relative page number of the last contiguous meta-data page for the table space. This value is only valid for DMS table spaces. It is 0 otherwise.

*Table 1798. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE                   |

---

## **tbsp\_max\_page\_top - Maximum table space page high watermark monitor element**

The highest allocated page number for a DMS table space since the database was activated.

*Table 1799. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | ACTIVITY METRICS BASE                   |

### **Usage**

This value changes whenever the value of the **tablespace\_page\_top** monitor element increases.

---

## **tbsp\_names - Table space names**

This element lists the table space names that the utility acts on.

*Table 1800. Event Monitoring Information*

| <b>Event Type</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-------------------|------------------------------|-----------------------|
| Change History    | UTILSTART                    | Always collected      |

### **Usage**

For the change history event monitor, if the **object\_type** element is DATABASE or TABLESPACE, this is a comma delimited list of table space names that the utility acts on.

---

## **tbsp\_trackmod\_state - Table space trackmod state monitor element**

The modification state that a table space is in with respect to the last or next backup.

*Table 1801. Table function monitoring information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b> |
|--|---|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | Always                                  |

## Usage

You can use this monitor element to determine the modification status of a table space. The status of a table space can be in one of the following states:

### CLEAN

No modifications occurred in the table space since the previous backup. If an incremental or delta backup is executed at this time, no data pages from this table space would be backed up.

### DIRTY

Table space contains data that needs to be picked up by the next backup.

### ININCREMENTAL

Table space contains modifications that were copied into an incremental backup. This state is in a DIRTY state relative to a full backup such that a future incremental backup needs to include some pages from this pool. This state is also in a CLEAN state such that a future delta backup does not need to include any pages from this pool.

### READFULL

The latest table space modification state change was caused by a dirty table space that is being read by a full backup that might not have completed successfully, or is currently in progress.

### READINCREMENTAL

The latest table space modification state change was caused by a dirty table space that is being read by an incremental backup that might not have completed successfully, or is currently in progress.

### UNAVAILABLE

The **trackmod** configuration parameter is set to No. Therefore, no table space modification status information is available.

---

## tcpip\_recv\_volume - TCP/IP received volume monitor element

The amount of data received by the data server from clients over TCP/IP. This value is reported in bytes.

*Table 1802. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1802. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1803. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## tcPIP\_recv\_wait\_time - TCP/IP received wait time monitor element

The time spent waiting for an incoming client request over TCP/IP excluding idle time. The value is given in milliseconds.

Table 1804. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1804. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1805. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## tcpip\_recvs\_total - TCP/IP receives total monitor element

The number of times data was received by the database server from the client application over TCP/IP.

Table 1806. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1806. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1807. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## tcPIP\_send\_volume - TCP/IP send volume monitor element

The amount of data sent by data server to client. This value is reported in bytes.

Table 1808. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1809. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## tcpip\_send\_wait\_time - TCP/IP send wait time monitor element

Time spent blocking on a TCP/IP send to the client. The value is given in milliseconds.

Table 1810. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1811. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | -                    |

## tcpip\_sends\_total - TCP/IP sends total monitor element

The number of times data was sent from the database server to the client application over TCP/IP.

Table 1812. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1813. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | Always collected     |

---

## temp\_tablespace\_top - Temporary table space top monitor element

The high watermark in KB for the temporary table space usage of DML activities at all nesting levels in a service class or work class. For service classes, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. For workloads, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE.

For service classes, when you remap activities between service subclasses with a REMAP ACTIVITY action, only the temp\_tablespace\_top high watermark of the service subclass where an activity completes is changed. High watermarks of service subclasses an activity is mapped to but does not complete in are unaffected.

Table 1814. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |
| Statistics | event_wlstats         | -              |

### Usage

Use this element to determine the highest DML activity system temporary table space usage reached on a member for a service class, workload, or work class in the time interval collected.

This element is only updated by activities that have a temporary table space threshold applied to them. If no temporary table space threshold is applied to an activity, a value of 0 is returned.

---

## territory\_code - Database Territory Code

The territory code of the database for which the monitor data is collected. This monitor element was formerly known as country\_code.

Table 1815. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Application    | appl                  | Basic          |

Table 1816. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch   |
|------------------|-----------------------|------------------|
| Event Log Header | event_log_header      | Always collected |
| Connections      | event_connheader      | Always collected |

**Usage** Territory code information is recorded in the database configuration file.

For DRDA AS connections, this element will be set to 0.

## thresh\_violations - Number of threshold violations monitor element

Number of times a threshold was violated.

This monitor element is an alias of the “num\_threshold\_violations - Number of threshold violations monitor element” on page 981 monitor element, which is returned by snapshot monitoring routines and the Database event monitor.

*Table 1817. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1818. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |

Table 1818. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## Usage

Use this element to quickly determine if there have been any WLM thresholds that have been violated. If thresholds have been violated you can then use the threshold violations event monitor (if created and active) to obtain details about the threshold violations.

For example, to obtain details which threshold was violated.

---

## threshold\_action - Threshold action monitor element

The action of the threshold to which this threshold violation record applies. Possible values include Stop, Continue and Remap.

Table 1819. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

## Usage

Use this element to determine whether the activity that violated the threshold was stopped when the violation occurred, was allowed to continue executing, or was remapped to another service subclass. If the activity was stopped, the application that submitted the activity will have received an SQL4712N error. If the activity was remapped to another service subclass, agents working for the activity on the member will be moving to the target service subclass of the threshold.

---

## threshold\_domain - Threshold domain monitor element

The domain of the threshold responsible for this queue.

Possible values are

- Database
- Work Action Set
- Service Superclass
- Service Subclass
- Workload

Table 1820. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

Table 1821. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |

## Usage

This element can be used for distinguishing the queue statistics of thresholds that have the same predicate but different domains.

---

## threshold\_maxvalue - Threshold maximum value monitor element

For non-queuing thresholds, this monitor element represents the value that was exceeded to cause this threshold violation. For queuing thresholds, this monitor element represents the level of concurrency that caused the queuing. The level of concurrency that caused the violation of the queuing threshold is the sum of **threshold\_maxvalue** and **threshold\_queuesize** monitor elements.

Table 1822. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch   |
|----------------------|---------------------------|------------------|
| Threshold violations | event_thresholdviolations | Always collected |

## Usage

For activity thresholds, this element provides a historical record of what the threshold's maximum value was at the time the threshold was violated. This is useful when the threshold's maximum value has changed since the time of the violation and the old value is no longer available from the SYSCAT.THRESHOLDS view. For the DATATAGINSC IN and DATATAGINSC NOT IN thresholds, this element contains the value of the data tag that violated the threshold.

---

## threshold\_name - Threshold name monitor element

The unique name of the threshold responsible for this queue.

Table 1823. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

Table 1824. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |

## Usage

Use this element to uniquely identify the queuing threshold whose statistics this record represents.

---

## threshold\_predicate - Threshold predicate monitor element

Identifies the type of threshold that was violated or for which statistics were collected.

*Table 1825. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics | ACTIVITY METRICS BASE            |

*Table 1826. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch   |
|----------------------|---------------------------|------------------|
| Threshold violations | event_thresholdviolations | Always collected |
| Statistics           | event_qstats              | Always collected |

### Usage

Use this monitor element in conjunction with other statistics or threshold violation monitor elements for analysis of a threshold violation.

The valid values for this monitor element when reported in the event\_thresholdviolations logical group are:

- AggSQLTempSpace
- SQLTempSpace
- SQLRowsReturned
- ActivityTotalTime
- EstimatedSQLCost
- TotalMemberConnections
- ConnectionIdleTime
- ConcurrentWorkloadOccurrences
- ConcurrentWorkloadActivities
- ConcurrentDBCoordActivities
- TotalSCMemberConnections
- SQLRowsRead
- SQLRowsReadInSC
- CPUTime
- CPUTimeInSC
- UowTotalTime
- DataTagInSC
- DataTagNotInSC

The valid values for this monitor element when reported in the event\_qstats logical group are:

- TotalMemberConnections

ConcurrentDBCoordActivities  
TotalSCMemberConnections

---

## threshold\_queuesize - Threshold queue size monitor element

The size of the queue for a queuing threshold. An attempt to exceed this size causes a threshold violation. For a non-queuing threshold, this value is 0.

Table 1827. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

### Usage

Use this element to determine the number of activities or connections in the queue for this threshold at the time the threshold was violated.

---

## thresholdid - Threshold ID monitor element

Identifies the threshold to which a threshold violation record applies or for which queue statistics were collected.

Table 1828. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_QUEUE_STATS table function -<br>Return threshold queue statistics | ACTIVITY METRICS BASE            |

Table 1829. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |
| Statistics           | event_qstats              | -              |

### Usage

Use this monitor element in conjunction with other activity history monitor elements for analysis of a threshold queue or for analysis of the activity that violated a threshold.

---

## time\_completed - Time completed monitor element

The time at which the activity described by this activity record finished executing. This element is a local timestamp.

Table 1830. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

## Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity.

This field has a value of "0000-00-00-00.00.00.000000" when a full activity record could not be written to a table event monitor due to memory limitations. If the activity was captured while it was in progress, then this field represents the time that activity was collected.

---

## time\_created - Time created monitor element

The time at which a user submitted the activity described by this activity record. This element is a local timestamp.

*Table 1831. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity.

---

## time\_of\_violation - Time of violation monitor element

The time at which the threshold violation described in this threshold violation record occurred. This element is a local timestamp.

*Table 1832. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

## Usage

Use this element in conjunction with other threshold violations monitor elements for analysis of a threshold violation.

---

## time\_stamp - Snapshot Time

The date and time when the database system monitor information was collected.

*Table 1833. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

**Usage** You can use this element to help relate data chronologically if you are saving the results in a file or database for ongoing analysis.

---

## time\_started - Time started monitor element

The time at which the activity described by this activity record began executing. This element is a local timestamp.

Table 1834. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity.

If the activity got rejected, then the value of **act\_exec\_time** monitor element is 0. In this case, the value of **time\_started** monitor element equals the value of **time\_completed** monitor element.

---

## time\_zone\_disp - Time Zone Displacement

Number of seconds that the local time zone is displaced from Greenwich Mean Time (GMT).

Table 1835. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

**Usage** All time reported by reported by the database system monitor is GMT, this displacement calculates the local time.

---

## top - Histogram bin top monitor element

The inclusive top end of the range of a histogram bin. The value of this monitor element is also the bottom exclusive end of the range of the next histogram bin.

Table 1836. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

### Usage

Use this element with the corresponding **bottom** element to determine the range of a bin within a histogram.

---

## tot\_log\_used\_top - Maximum Total Log Space Used

The maximum amount of total log space used (in bytes).

Table 1837. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TRANSACTION_LOG table function - Get log information | ACTIVITY METRICS BASE            |

Table 1838. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 1839. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You can use this element to help evaluate the amount of primary log space that you have allocated. Comparing the value of this element with the amount of primary log space you have allocated can help you to evaluate your configuration parameter settings. Your primary log space allocation can be calculated using the following formula:

$$\text{logprimary} \times \text{logfilsiz} \times 4096 \text{ (see note below)}$$

You can use this element in conjunction with *sec\_log\_used\_top* and *sec\_logs\_allocated* to show your current dependency on secondary logs.

This value includes space used in both primary and secondary log files.

You may need to adjust the following configuration parameters:

- logfilsiz
- logprimary
- logsecond

**Note:** While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

---

## total\_act\_time - Total activity time monitor element

The total amount of time spent executing activities. This value is given in milliseconds.

Table 1840. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 1840. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 1841. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

Use this monitor element along with the **total\_act\_wait\_time** monitor element to determine the percentage of time the data server spent working on the activity.

$$\frac{(\text{total\_act\_time} - \text{total\_act\_wait\_time})}{(\text{total\_act\_time})} = \%$$

% of time data server is actively working on activity

## total\_act\_wait\_time - Total activity wait time monitor element

Total time spent waiting within the DB2 database server, while processing an activity. The value is given in milliseconds.

Table 1842. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1842. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1843. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

Use this monitor element along with the **total\_act\_time** monitor element to determine the percentage of time the data server spent working on the activity.

$$\frac{(\text{total\_act\_time} - \text{total\_act\_wait\_time})}{(\text{total\_act\_time})} =$$

% of time data server is actively working on activity

---

## total\_app\_commits - Total application commits monitor elements

Total number of commit statements issued by the client application.

*Table 1844. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1845. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_app\_rollbacks - Total application rollbacks monitor element

Total number of rollback statements issued by the client application.

Table 1846. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1847. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

---

## total\_app\_rqst\_time - Total application request time monitor element

The total elapsed time spent on application requests; this is the total time spent by coordinator agents on the server executing application requests. This value is reported in milliseconds.

*Table 1848. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

### Usage

Use this monitor element to determine the time that the application request spent in the DB2 data server. This value can be used to help determine if the data server is the source of an observed performance problem.

For example, if a user reports that there is a problem with an application and it has taken 20 minutes to return, and if you determine that total application request time is 1 minute and there are currently no application requests in progress for the connection, then the performance problem might lie outside of the DB2 data server.

## total\_app\_section\_executions - Total application section executions monitor element

Number of section executions performed by an application.

Table 1849. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1850. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 1850. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |

## total\_buffers\_rcvd - Total FCM Buffers Received

For snapshot monitor, this monitor element reports the total number of FCM buffers received by the node issuing the GET SNAPSHOT command from the node identified by the **node\_number** monitor element. For table function monitor, this monitor element reports the total number of FCM buffers received from a remote database member.

Table 1851. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

Table 1852. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

### Usage

Use this element to measure the level of traffic between the current member and the remote member. If the total number of FCM buffers received from this member is high, consider redistributing the database or moving tables to reduce the traffic between members.

## total\_buffers\_sent - Total FCM Buffers Sent

For snapshot monitor, this monitor element reports the total number of FCM buffers that have been sent from the node issuing the GET SNAPSHOT command to the node identified by the **node\_number** monitor element. For table function monitor, this monitor element reports the total number of FCM buffers sent from the current database member to a remote database member.

Table 1853. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections | ACTIVITY METRICS BASE            |

Table 1854. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

## Usage

Use this element to measure the level of traffic between the current member and the remote member. If the total number of FCM buffers sent to this member is high, consider redistributing the database or moving tables to reduce the traffic between members.

---

### total\_bytes\_received - Bytes received monitor element

Total number of bytes received since the network adapter started.

*Table 1855. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

### total\_bytes\_sent - Bytes sent monitor element

Total number of bytes sent since the network adapter started.

*Table 1856. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |

---

### total\_commit\_proc\_time - Total commits processing time monitor element

The total amount of processing (non-wait) time spent performing commit processing on the database server. The value is given in milliseconds.

*Table 1857. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1857. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1858. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_commit\_time - Total commit time monitor element

The total amount of time spent performing commit processing on the database server. The value is given in milliseconds.

Table 1859. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1859. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1860. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_compilations - Total compilations monitor element

The total number of explicit compiles on the database server. Explicit compiles are compilations directly initiated by a user request such as a bind, rebind, prepare or execute immediate.

Table 1861. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1861. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1862. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_compile\_proc\_time - Total compile processing time monitor element

The total amount of processing (non-wait) time spent performing explicit compiles on the database server. Explicit compiles are compilations directly initiated by a user request such as a bind, rebind, prepare or execute immediate. The value is given in milliseconds.

Table 1863. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1863. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1864. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_compile\_time - Total compile time monitor element

The total amount of time spent performing explicit compiles on the database server. Explicit compiles are compilations directly initiated by a user request such as a bind, rebind, prepare or execute immediate. The value is given in milliseconds.

Table 1865. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - Get formatted row-based component times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1865. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1866. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_cons - Connects Since Database Activation

Indicates the number of connections to the database since the first connect, activate, or last reset (coordinator agents).

Table 1867. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1868. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |

**Usage** You can use this element in conjunction with the db\_conn\_time and the db2start\_time monitor elements to calculate the frequency at which applications have connected to the database.

If the frequency of connects is low, you may want to explicitly activate the database using the ACTIVATE DATABASE command before connecting any other application, because of the extra processing time that is associated with the first connect to a database (for example, initial buffer pool allocation). This will result in subsequent connects being processed at a higher rate.

**Note:** When you reset this element, its value is set to the number of applications that are currently connected, not to zero.

## total\_connect\_authentication\_proc\_time - Total connection authentication processing time monitor element

The amount of processing (non-wait) time spent performing connection or switch user authentication, in milliseconds.

Table 1869. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1870. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of Work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_connect\_authentications - Connections or switch user authentications performed monitor element

The number of connection or switch user authentications performed.

Table 1871. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1872. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## total\_connect\_authentication\_time - Total connection or switch user authentication request time monitor element

The amount of time spent performing connection or switch user authentication, in milliseconds.

Table 1873. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1874. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## total\_connect\_request\_proc\_time - Total connection or switch user request processing time monitor element

The amount of processing (non-wait) time spent processing a connection or switch user request, in milliseconds.

Table 1875. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1876. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document)     | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## total\_connect\_requests - Connection or switch user requests monitor element

The total number of connection or switch user requests.

*Table 1877. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1878. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## total\_connect\_request\_time - Total connection or switch user request time monitor element

The amount of time spent performing a connection or switch user request, in milliseconds.

Table 1879. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1880. Event Monitoring Information

| Event Type   | Logical Data Grouping                                     | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document)     | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)      | REQUEST METRICS BASE |
| Unit of Work | uow (reported in the metrics.xml document)<br>uow_metrics | REQUEST METRICS BASE |

## total\_cpu\_time - Total CPU time monitor element

The total amount of CPU time used while within DB2. Represents total of both user and system CPU time. This value is given in microseconds.

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS table function, it represents the total CPU time since the last reset of statistics. When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS table function, it represents the total CPU time since the function was executed.

*Table 1881. Table Function Monitoring Information*

| <b>Table Function</b>  | <b>Monitor Element Collection Level</b>  |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics  | REQUEST METRICS BASE   |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample workload metrics  | REQUEST METRICS BASE   |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                                    | REQUEST METRICS BASE   |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities  | REQUEST METRICS BASE   |

Table 1881. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics | REQUEST METRICS BASE             |

Table 1882. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Together with the **total\_disp\_run\_queue\_time** monitor element, **total\_cpu\_time** can be used to compute a measure of the amount of contention for the CPU resource, measured on a scale from 0 to 1, with lower numbers meaning greater contention for the CPU resource. This measure, called CPU velocity, is computed by measuring the amount of time that work in the service class has access to the CPU divided by the total time spent accessing the CPU or waiting to access the CPU. CPU velocity gives a measure of how efficiently the work executing in the service class is being executed relative to how efficiently it could be executed if such work never had to wait for the CPU. The formula is as follows:

$$\text{CPU velocity} = \text{total\_cpu\_time} / (\text{total\_cpu\_time} + \text{total\_disp\_run\_queue\_time})$$

---

## total\_disp\_run\_queue\_time - Total dispatcher run queue time monitor element

The total time that requests, that were run in this service class, spent waiting to access the CPU. This value is given in microseconds.

Table 1883. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 1883. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element  | Collection Level |
|--|------------------|------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS  | BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries  | ACTIVITY METRICS | BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS  | BASE             |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics  | REQUEST METRICS  | BASE             |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample workload metrics  | REQUEST METRICS  | BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                                    | REQUEST METRICS  | BASE             |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics   | REQUEST METRICS  | BASE             |

Table 1884. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document             | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document               | REQUEST METRICS BASE  |

## Usage

Together with the **total\_cpu\_time**, the **total\_disp\_run\_queue\_time** monitor element can be used to compute a measure of the amount of contention for the CPU resource, measured on a scale from 0 to 1, with lower numbers meaning greater contention for the CPU resource. This measure, called CPU velocity, is computed by measuring the amount of time that work in a service class has access to the CPU divided by the total time spent accessing the CPU or waiting to access the CPU. It gives a measure of how efficiently the work is being executed relative to how efficiently it could be executed if such work never had to wait for the CPU. The formula is as follows:

$$\text{CPU velocity} = \text{total\_cpu\_time} / (\text{total\_cpu\_time} + \text{total\_disp\_run\_queue\_time})$$

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS function, this monitor element represents the total dispatcher run queue wait time since the last reset of statistics.

When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS function, this monitor element represents the total dispatcher run queue wait time since the function was executed.

---

## total\_exec\_time - Elapsed statement execution time monitor element

The total time in seconds and microseconds that was spent executing a particular statement in the SQL cache.

Table 1885. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element with **num\_executions** monitor element determine the average elapsed time for the statement and identify the SQL statements that would most benefit from a tuning of their SQL. The **num\_compilation** monitor element must be considered when evaluating the contents of this element.

**Note:** Due to the differences in granularity with which the DB2 system collects statistics, the value of the **total\_exec\_time** monitor element might not equal the sum of values of **system\_cpu\_time** and **user\_cpu\_time** monitor elements. In this case, the sum of **system\_cpu\_time** and **user\_cpu\_time** monitor elements more accurately reflects the actual total execution time.

## total\_extended\_latch\_wait\_time - Total extended latch wait time monitor element

The amount of time, in milliseconds, spent in extended latch waits.

Table 1886. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level  |
|---|---|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML documented provided as input. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                                | Not applicable; reports whichever elements are contained in the XML documented provided as input. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times  | Not applicable; reports whichever elements are contained in the XML documented provided as input. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE  |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE  |
| MON_GET_EXTENDED_LATCH_WAITS  | REQUEST METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE   |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE  |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE  |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE  |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE  |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE  |

Table 1887. Event Monitoring Information

| Event Type    | Logical Data Grouping   | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document)               | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics   | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Unit of work  | uow (reported in the metrics.xml document)<br>uow_metrics           | REQUEST METRICS BASE  |
| Package cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics | ACTIVITY METRICS BASE |

## Usage

- Use the following formula to determine extended latch wait time as a percentage of total wait time. This formula can be used to determine if the time spent waiting on extended latches is high relative to the total wait time.  

$$(TOTAL\_EXTENDED\_LATCH\_WAIT\_TIME / TOTAL\_WAIT\_TIME) * 100$$
- Use the following formula to determine the average length of time in milliseconds of an extended latch wait.  

$$TOTAL\_EXTENDED\_LATCH\_WAIT\_TIME / TOTAL\_EXTENDED\_LATCH\_WAITS$$

## total\_extended\_latch\_waits - Total extended latch waits monitor element

The number of extended latch waits.

Table 1888. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 1888. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |
| MON_GET_EXTENDED_LATCH_WAITS   | REQUEST METRICS BASE             |

Table 1889. Event Monitoring Information

| Event Type    | Logical Data Grouping   | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document)               | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics   | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)                | REQUEST METRICS BASE  |
| Unit of work  | uow (reported in the metrics.xml document)<br>uow_metrics           | REQUEST METRICS BASE  |
| Package cache | pkgcache (reported in the metrics.xml document)<br>pkgcache_metrics | ACTIVITY METRICS BASE |

## Usage

Use the following formula to determine the average length of time in milliseconds of an extended latch wait.

$$\text{TOTAL\_EXTENDED\_LATCH\_WAIT\_TIME} / \text{TOTAL\_EXTENDED\_LATCH\_WAITS}$$

---

## total\_move\_time - Total extent move time monitor element

In milliseconds, the total move time for all extents moved during the table space rebalance process.

*Table 1890. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_EXTENT_MOVEMENT_STATUS<br>- Get extent movement progress status metrics | ACTIVITY METRICS BASE            |

---

---

## total\_hash\_joins - Total Hash Joins

The total number of hash joins executed.

**Element identifier**

total\_hash\_joins

**Element type**

counter

*Table 1891. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1892. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** At the database or application level, use this value in conjunction with hash\_join\_overflows and hash\_join\_small\_overflows to determine if a significant percentage of hash joins would benefit from modest increases in the sort heap size.

---

## total\_hash\_loops - Total Hash Loops

The total number of times that a single partition of a hash join was larger than the available sort heap space.

*Table 1893. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1894. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** Values for this element indicate inefficient execution of hash joins. This might indicate that the sort heap size is too small or the sort heap threshold is too small. Use this value in conjunction with the other hash join variables to tune the sort heap size (*sortheap*) and sort heap threshold (*sheapthres*) configuration parameters.

---

## total\_implicit\_compilations - Total implicit complications monitor element

Total number of implicit compiles on the database server. Implicit compiles are compilations that are not directly requested by the user. That is, they are not a result of a bind, rebind, prepare or execute immediate request. For example, an implicit compilation may occur when executing a statement that was bound using the VALIDATE RUN option if the statement needs to be compiled at execution time.

Table 1895. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1896. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_implicit\_compile\_proc\_time - Total implicit compile processing time monitor element

Total amount of processing (non-wait) time spent performing implicit compiles on the database server. Implicit compiles are compilations that are not directly requested by the user. That is, they are not a result of a bind, rebind, prepare or execute immediate request. For example, an implicit compilation may occur when executing a statement that was bound using the VALIDATE RUN option if the statement needs to be compiled at execution time. The value is given in milliseconds.

Table 1897. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1898. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_implicit\_compile\_time - Total implicit compile time monitor element

The total amount of time spent performing implicit compiles on the database server. Implicit compiles are compilations that are not directly requested by the user. That is, they are not a result of a bind, rebind, prepare or execute immediate request. For example, an implicit compilation may occur when executing a statement that was bound using the VALIDATE RUN option if the statement needs to be compiled at execution time. The value is given in milliseconds.

Table 1899. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1900. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_load\_proc\_time - Total load processing time monitor element

Total amount of processing (non-wait) time spent performing load processing on the database server. The value is given in milliseconds.

Table 1901. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1902. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch       |
|------------|---|----------------------|
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |

Table 1902. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_load\_time - Total load time monitor element

The total amount of time spent performing loads on the database server. The value is given in milliseconds.

Table 1903. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1904. Event Monitoring Information

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |

Table 1904. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                    | Monitor Switch       |
|--------------|--|----------------------|
| Unit of work | Reported in the system_metrics document. | REQUEST METRICS BASE |

## total\_loads - Total loads monitor element

The total number of loads performed on the database server.

Table 1905. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1906. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

---

## total\_log\_available - Total Log Available

The amount of active log space in the database that is not being used by uncommitted transactions (in bytes).

*Table 1907. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1908. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

### Usage

Use this element in conjunction with total\_log\_used to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- logfilesiz
- logprimary
- logsecond

If total\_log\_available goes down to 0, SQL0964N will be returned. You may need to increase the previously listed configuration parameters, or end the oldest transaction by COMMIT, ROLLBACK or FORCE APPLICATION.

If logsecond is set to -1 this element will contain SQLM\_LOGSPACE\_INFINITE.

**Note:** While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

---

## total\_log\_used - Total Log Space Used

The total amount of active log space currently used (in bytes) in the database.

*Table 1909. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TRANSACTION_LOG table<br>function - Get log information | ACTIVITY METRICS BASE            |

*Table 1910. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** Use this element in conjunction with total\_log\_available to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- logfilesiz
- logprimary

- logsecond

**Note:** While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

---

## total\_move\_time - Total extent move time monitor element

In milliseconds, the total move time for all extents moved during the table space rebalance process.

*Table 1911. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_EXTENT_MOVEMENT_STATUS<br>- Get extent movement progress status metrics | ACTIVITY METRICS BASE            |

---

## total\_olap\_funcs - Total OLAP Functions monitor element

The total number of OLAP functions executed.

*Table 1912. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 1913. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

### Usage

At the database or application level, use this value in conjunction with `olap_func_overflows` to determine if a significant percentage of OLAP functions would benefit from modest increases in the sort heap size.

---

## total\_peas - Total partial early aggregations monitor element

The total number of times that partial early aggregation operations have been executed.

*Table 1914. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details        | ACTIVITY METRICS BASE  |

Table 1914. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_CONNECTION table function - Get connection metrics                                      | REQUEST METRICS BASE             |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics                     | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics         | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                 | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                         | REQUEST METRICS BASE             |

Table 1915. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |

Table 1915. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element | Collection Level |
|---|-----------------|------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK<br>table function - Get unit of work metrics   | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD<br>table function - Get workload metrics   | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS<br>table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS | BASE             |

Table 1916. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Connection    | event_conn  | -                     |
| Statements    | event_stmt  | -                     |
| Transactions  | event_xact  | -                     |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

At the database or application level, use this value in conjunction with **post\_threshold\_peas** to determine if a significant number of partial early aggregation operations would benefit from an increase in either sort heap size or sort heap threshold. If the ratio of **post\_threshold\_peas** to **total\_peas** is high, increasing the sort heap size or the sort heap threshold, or both, may improve database or application performance.

## total\_peds - Total partial early distincts monitor element

The total number of times that partial early distinct operations have been executed.

Table 1917. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1918. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

Table 1918. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Connection    | event_conn                                 | -                     |
| Statements    | event_stmt                                 | -                     |
| Transactions  | event_xact                                 | -                     |
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |

## Usage

At the database or application level, use this value in conjunction with the **disabled\_peds** monitor element and the **post\_threshold\_peds** monitor element to determine if a significant number of partial early distinct operations would benefit from an increase in either sort heap size or sort heap threshold. If the ratio of the **disabled\_peds** monitor element and the **post\_threshold\_peds** monitor element to the **total\_peds** monitor element is high, increasing the sort heap size or the sort heap threshold, or both, may improve database or application performance.

---

## total\_reorg\_proc\_time - Total reorganization processing time monitor element

The total amount of processing (non-wait) time spent performing reorg operations on the database server. The value is given in milliseconds.

Table 1919. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

Table 1919. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1920. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_reorg\_time - Total reorganization time monitor element

The total amount of time spent performing reorg operations on the database server. The value is given in milliseconds.

Table 1921. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1922. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_reorgs - Total reorganizations monitor element

The number of reorg operations issued against the database server.

Table 1923. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1924. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_rollback\_proc\_time - Total rollback processing time monitor element

The total amount of processing (non-wait) time spent performing rollback operations on the database server. The value is given in milliseconds.

*Table 1925. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1926. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_rollback\_time - Total rollback time monitor element

The total amount of time spent performing rollback operations on the database server. The value is given in milliseconds.

*Table 1927. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1928. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_routine\_invocations - Total routine invocations monitor elements

The total number of times a routine was invoked.

*Table 1929. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1930. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 1930. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                      | Monitor Switch        |
|---------------|--|-----------------------|
| Package cache | Reported in the activity_metrics document. | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.   | REQUEST METRICS BASE  |

## total\_routine\_non\_sect\_proc\_time - Non-section processing time monitor element

The total amount of processing time this statement spent performing non-section execution within routines. This value includes both the time spent executing user-code within routines and time spent performing non-section operations like commit or rollback. Processing time does not include wait time. The value is given in milliseconds.

Table 1931. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries             | ACTIVITY METRICS BASE  |

Table 1932. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## total\_routine\_non\_sect\_time - Non-section routine execution time monitor elements

The total amount of time this statement spent performing non-section execution within routines. This value includes both the time spent executing user-code within routines and the time spent performing non-section operations like commit or rollback. The value is given in milliseconds.

Table 1933. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based<br>component times                           | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics                             | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries             | ACTIVITY METRICS BASE  |

Table 1934. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## total\_routine\_time - Total routine time monitor element

The total time spent executing routines. The value is given in milliseconds.

Table 1935. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based<br>component times                               | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics                                 | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function -<br>Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                | ACTIVITY METRICS BASE  |

Table 1935. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element  | Collection Level |
|---|------------------|------------------|
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS | BASE             |
| MON_GET_SERVICE_SUBCLASS<br>table function - Get service subclass metrics   | REQUEST METRICS  | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK<br>table function - Get unit of work metrics   | REQUEST METRICS  | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD<br>table function - Get workload metrics   | REQUEST METRICS  | BASE             |
| MON_GET_WORKLOAD_DETAILS<br>table function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS  | BASE             |

Table 1936. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |

## Usage

If the collection level is set to BASE, then the value of **total\_routine\_time** monitor element does not include any time spent executing functions that were defined using the NO SQL clause.

If the collection level is set to EXTENDED, then the value of **total\_routine\_time** monitor element includes the time spent in all routines.

## total\_routine\_user\_code\_proc\_time - Total routine user code processing time monitor element

The total amount of processing time spent executing in routines outside of known DB2 times (typically user code in routines). The value is given in milliseconds.

Table 1937. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1938. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 1938. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## Usage

If the collection level is set to BASE, then this monitor element does not include any processing time spent executing functions that were defined using the NO SQL clause. Instead, this time is included in the value of the **total\_section\_proc\_time** monitor element.

If the collection level is set to EXTENDED, then the value of this monitor element includes the processing time spent executing all routines.

---

## total\_routine\_user\_code\_time - Total routine user code time monitor element

The total amount of time spent executing in routines outside of known DB2 times (typically user code in routines). The value is given in milliseconds.

Table 1939. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times                            | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 1939. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element | Collection Level |
|---|-----------------|------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS | BASE             |

Table 1940. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |

## Usage

If the collection level is set to BASE, then the value of this monitor element does not include the time spent executing functions that were defined using the NO SQL clause. Instead, this time is included in the value of the **total\_section\_time** monitor element.

If the collection level is set to EXTENDED, then the value of this monitor element includes the time spent executing all routines.

---

## total\_rqst\_mapped\_in - Total request mapped-in monitor element

The total number of requests that were mapped into this service subclass via a remap threshold or a work action set.

*Table 1941. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

*Table 1942. Event Monitoring Information*

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |

---

## total\_rqst\_mapped\_out - Total request mapped-out monitor element

The total number of requests that were mapped out of this service subclass via a remap threshold or a work action set.

*Table 1943. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

*Table 1944. Event Monitoring Information*

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |

---

## total\_rqst\_time - Total request time monitor element

The total amount of time spent working on requests. This value is reported in milliseconds.

*Table 1945. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - Get formatted row-based component times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

Table 1945. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_TIMES_BY_ROW -<br>Get formatted row-based combined hierarchy<br>wait and processing times                            | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE   |

Table 1946. Event Monitoring Information

| Event Type   | Logical Data Grouping                                   | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the<br>details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the<br>details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the<br>system_metrics document.             | REQUEST METRICS BASE |

## total\_runstats - Total runtime statistics monitor element

The total number of runstats operations performed on the database server.

Table 1947. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_CONNECTION table function -<br>Get connection metrics                        | REQUEST METRICS BASE   |

Table 1947. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1948. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_runstats\_proc\_time - Total runtime statistics processing time monitor element

The total amount of processing (non-wait) time spent performing runstats operations on the database server. The value is given in milliseconds. Any time the runstats utility spends throttled does not count to the runstats processing time.

Table 1949. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 1949. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1950. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## **total\_runstats\_time - Total runtime statistics time monitor element**

The total amount of time spent performing runstats operations on the database server. The value is given in milliseconds.

Table 1951. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - Get formatted row-based component times                                | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1951. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1952. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch       |
|--------------|---|----------------------|
| Statistics   | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.              | REQUEST METRICS BASE |

## total\_sec\_cons - Secondary Connections

The number of connections made by a subagent to the database at the node.

### Element identifier

total\_sec\_cons

### Element type

counter

Table 1953. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** You can use this element in conjunction with the total\_cons, db\_conn\_time, and the db2start\_time monitor elements to calculate the frequency at which applications have connected to the database.

## total\_section\_proc\_time - Total section processing time monitor element

The total amount of processing time agents spent performing section execution. Processing time does not include wait time. The value is given in milliseconds.

Table 1954. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 1955. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 1955. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

If the collection level is set to BASE, then the value of the **total\_section\_proc\_time** monitor element includes processing time spent executing functions that were defined using the NO SQL clause.

If the collection level is set to EXTENDED, then the processing time spent executing these functions is not included in the value of the **total\_section\_proc\_time** monitor element. It is included in the value of the **total\_routine\_user\_code\_proc\_time** monitor element.

---

## total\_section\_sort\_proc\_time - Total section sort processing time monitor element

Total amount of processing (non-wait) time spent performing sorts while executing a section, which is the execution of the compiled query plan generated by the SQL statement that was issued by the client application. The value is given in milliseconds.

Table 1956. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |

Table 1956. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1957. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

At the system level, use this element with the **total\_section\_sorts** monitor element to calculate the average sort processing time (does not include waits) during section execution, which can indicate whether or not sorting is an issue as far as performance is concerned.

At the activity level, use this element to identify statements that spend a large amount of time sorting. These statements may benefit from additional tuning to reduce the sort time.

## total\_section\_sort\_time - Total section sort time monitor element

Total amount of time spent performing sorts while executing a section, which is the execution of the compiled query plan generated by the SQL statement that was issued by the client application. The value is given in milliseconds.

Table 1958. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document)                    | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE   |

Table 1959. Event Monitoring Information

| Event Type | Logical Data Grouping                                    | Monitor Switch        |
|------------|--|-----------------------|
| Activities | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                    | ACTIVITY METRICS BASE |

Table 1959. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## Usage

At the system level, use this element with the **total\_section\_sorts** monitor element to calculate the average sort time during section execution, which can indicate whether or not sorting is an issue as far as statement performance is concerned.

The **total\_section\_sort\_time** element includes both wait and processing time. If the value of (total\_section\_sort\_time - total\_section\_sort\_proc\_time) is high, sorts are spending a lot of time waiting. For example, if sorts are frequently spilling to disk, the value of the **total\_section\_sort\_time** monitor element will increase due to I/O waits. This time will not be included in the **total\_section\_sort\_proc\_time** monitor element value, which only counts the time actively processing a sort. In this case, you may consider tuning sort memory to improve performance.

At the activity level, use this element to identify statements that spend a large amount of time sorting. These statements may benefit from additional tuning to reduce the sort time.

---

## total\_section\_sorts - Total section sorts monitor element

Total number of sorts performed during section execution, which is the execution of the compiled query plan generated by the SQL statement that was issued by the client application.

Table 1960. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 1960. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element       | Collection Level |
|--|-----------------------|------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE |                  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE |                  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE  |                  |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE  |                  |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE  |                  |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE  |                  |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE  |                  |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE  |                  |

Table 1961. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## Usage

Use this element with the **total\_section\_sort\_time** monitor element to calculate the average amount of time spent performing sorts during section execution.

At the activity and package cache levels, use this element to identify statements which are performing large numbers of sorts. These statements may benefit from additional tuning to reduce the number of sorts. You can also use the EXPLAIN statement to identify the number of sorts a statement performs.

---

## total\_section\_time - Total section time monitor element

The total time agents spent performing section execution. The value is given in milliseconds.

*Table 1962. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based<br>component times  | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW -<br>Get formatted row-based output for all<br>metrics  | Not applicable; reports whichever elements<br>are contained in the XML document<br>provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document)                    | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                               | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries                                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE   |

*Table 1963. Event Monitoring Information*

| Event Type | Logical Data Grouping                                    | Monitor Switch        |
|------------|--|-----------------------|
| Activities | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the<br>details_xml) document) | REQUEST METRICS BASE  |

Table 1963. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |

## Usage

If the collection level is set to BASE, then the value of **total\_section\_time** monitor element includes time spent executing functions that were defined using the NO SQL clause.

If the collection level is set to EXTENDED, then the time spent executing these functions is not included in the value of the **total\_section\_time** monitor element. It is included in the value of the **total\_routine\_user\_code\_time** monitor element instead.

---

## total\_sort\_time - Total sort time monitor element

The total elapsed time for all sorts that have been executed. This value is reported in milliseconds.

Table 1964. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Sort           |
| Application    | appl                  | Sort           |
| Application    | stmt                  | Sort           |
| Dynamic SQL    | dynsql                | Sort           |

For snapshot monitoring, this counter can be reset.

Table 1965. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |
| Statements | event_stmt            | Always collected |
| Activities | event_activity        | Statement, Sort  |

## Usage

At a database or application level, use this element with **total\_sorts** to calculate the average sort time, which can indicate whether or not sorting is an issue as far as performance is concerned.

At a statement level, use this element to identify statements that spend a lot of time sorting. These statements may benefit from additional tuning to reduce the sort time.

This count also includes sort time of temporary tables created during related operations. It provides information for one statement, one application, or all applications accessing one database.

When using monitor elements providing elapsed times, you should consider:

1. Elapsed times are affected by system load, so the more processes you have running, the higher this elapsed time value.
2. To calculate this monitor element at a database level, the database system monitor sums the application-level times. This can result in double counting elapsed times at a database level, since more than one application process can be running at the same time.

To provide meaningful data from the database level, you should normalize the data to a lower level. For example:

```
total_sort_time / total_sorts
provides information about the average elapsed time for each sort.
```

---

## total\_sorts - Total sorts monitor element

The total number of sorts that have been executed.

*Table 1966. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |

Table 1966. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element | Collection Level |
|--|-----------------|------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS | BASE             |

Table 1967. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1968. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Statements    | event_stmt  | Always collected      |
| Activities    | event_activity  | Statement, Sort       |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

At a database or application level, use this value with **sort\_overflows** to calculate the percentage of sorts that need more heap space. You can also use it with **total\_sort\_time** to calculate the average sort time.

If the number of sort overflows is small with respect to the total sorts, then increasing the sort heap size may have little impact on performance, unless this buffer size is increased substantially.

At a statement level, use this element to identify statements which are performing large numbers of sorts. These statements may benefit from additional tuning to reduce the number of sorts. You can also use the SQL EXPLAIN statement to identify the number of sorts a statement performs.

## total\_stats\_fabrication\_proc\_time - Total statistics fabrication processing time monitor element

The total non-wait time spent on statistics fabrications by real-time statistics gathering, in millisecond. Statistics fabrication is the statistics collection activity needed to generate statistics during query compilation.

*Table 1969. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1970. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_stats\_fabrication\_time - Total statistics fabrication time monitor element

The total time spent on statistics fabrications by real-time statistics gathering, in milliseconds. Statistics fabrication is the statistics collection activity needed to generate statistics during query compilation.

Table 1971. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1972. Event Monitoring Information

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Activities | event_activity                                       | Always collected     |
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |

Table 1972. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | uow_metrics  | REQUEST METRICS BASE |
| Package cache | pkgcache   | Always collected     |

## total\_stats\_fabrications - Total statistics fabrications monitor elements

The total number of statistics fabrications performed by real-time statistics gathering. Statistics fabrication is the statistics collection activity needed to generate statistics during query compilation.

Table 1973. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 1974. Event Monitoring Information

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity                                       | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | Always collected      |

## total\_sync\_runstats\_time - Total synchronous RUNSTATS time monitor elements

The total time spent on synchronous RUNSTATS activities triggered by real-time statistics gathering, in milliseconds. The synchronous RUNSTATS activities occur during query compilation.

Table 1975. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_COMPONENT<br>_TIMES_BY_ROW - Get formatted row-based component times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |

Table 1975. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 1976. Event Monitoring Information

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity                                       | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | Always collected      |

## total\_sync\_runstats\_proc\_time - Total synchronous RUNSTATS processing time monitor element

The non-wait time spent on synchronous RUNSTATS activities triggered by real-time statistics gathering, in milliseconds. The synchronous RUNSTATS activities occur during query compilation.

Table 1977. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 1977. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 1978. Event Monitoring Information

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## total\_sync\_runstats - Total synchronous RUNSTATS activities monitor element

The total number of synchronous RUNSTATS activities triggered by real-time statistics gathering. The synchronous RUNSTATS activities occur during query compilation.

Table 1979. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details   | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |

Table 1979. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 1980. Event Monitoring Information

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity                                       | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.           | Always collected      |

## total\_sys\_cpu\_time - Total system CPU time for a statement monitor element

The total system CPU time for an SQL statement.

Table 1981. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

### Usage

Use this element with Elapsed Statement Execution Time and Total User CPU for a Statement to evaluate which statements are the most expensive.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two

subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

## total\_sorts - Total sorts monitor element

The total number of sorts that have been executed.

*Table 1982. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1983. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 1984. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Statements    | event_stmt  | Always collected      |
| Activities    | event_activity  | Statement, Sort       |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

At a database or application level, use this value with **sort\_overflows** to calculate the percentage of sorts that need more heap space. You can also use it with **total\_sort\_time** to calculate the average sort time.

If the number of sort overflows is small with respect to the total sorts, then increasing the sort heap size may have little impact on performance, unless this buffer size is increased substantially.

At a statement level, use this element to identify statements which are performing large numbers of sorts. These statements may benefit from additional tuning to reduce the number of sorts. You can also use the SQL EXPLAIN statement to identify the number of sorts a statement performs.

---

## total\_usr\_cpu\_time - Total user CPU time for a statement monitor element

The total user CPU time for an SQL statement.

Table 1985. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Statement      |

For snapshot monitoring, this counter can be reset.

## Usage

Use this element with Elapsed Statement Execution Time and to evaluate the longest running statements.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

---

## total\_wait\_time - Total wait time monitor element

The total time spent waiting within the DB2 database server. The value is given in milliseconds.

*Table 1986. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 1987. Event Monitoring Information*

| Event Type | Logical Data Grouping                                | Monitor Switch       |
|------------|--|----------------------|
| Statistics | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |

Table 1987. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | Reported in the system_metrics document.             | REQUEST METRICS BASE |

## Usage

To understand the percentage of time the database server spends actively working on requests, use the following ratio:

$$(total\_rqst\_time - total\_wait\_time) / total\_rqst\_time$$

The value of the **client\_idle\_wait\_time** monitor element is not included in the value of the **total\_wait\_time** monitor element. The **total\_wait\_time** element represents only time spent waiting while the database server is processing requests.

---

## tpmon\_acc\_str - TP monitor client accounting string monitor element

The data passed to the target database for logging and diagnostic purposes, if the sqleseti API was issued in this connection. The current value of the CLIENT\_ACCTNG special register for this connection, unit of work, or activity.

This monitor element is synonymous to the **client\_acctng** monitor element. The **client\_acctng** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_acc\_str** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

Table 1988. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |

Table 1989. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Activities  | event_activity        | -              |
| Deadlock    | event_dlconn          | -              |
| Transaction | event_xact            | -              |

## Usage

Use this element for problem determination and accounting purposes.

---

## tpmon\_client\_app - TP monitor client application name monitor element

Identifies the server transaction program performing the transaction, if the sqleset API was issued in this connection. The current value of the CLIENT\_APPLNAME special register for this connection, unit of work, or activity.

This monitor element is synonymous to the **client\_applname** monitor element. The **client\_applname** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_app** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

*Table 1990. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |

*Table 1991. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Activities  | event_activity        | -              |
| Deadlock    | event_dlconn          | -              |
| Transaction | event_xact            | -              |

### Usage

Use this element for problem determination and accounting purposes.

---

## tpmon\_client\_userid - TP monitor client user ID monitor element

The client user ID generated by a transaction manager and provided to the server, if the sqleset API is used. The current value of the CLIENT\_USERID special register for this connection, unit of work, or activity.

This monitor element is synonymous to the **client\_userid** monitor element. The **client\_userid** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_userid** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

*Table 1992. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |

*Table 1993. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |
| Deadlock   | event_dlconn          | -              |

Table 1993. Event Monitoring Information (continued)

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Transaction | event_xact            | -              |

## Usage

Use this element in application server or Transaction Processing monitor environments to identify the end-user for whom the transaction is being executed.

---

## tpmon\_client\_wkstn - TP monitor client workstation name monitor element

Identifies the client's system or workstation (for example CICS EITERMID), if the sqleseti API was issued in this connection. The current value of the CLIENT\_WRKSTNNAME special register for this connection, unit of work, or activity.

This monitor element is synonymous to the **client\_wrkstnname** monitor element. The **client\_wrkstnname** monitor element is used for monitoring table functions and event monitors that write to unformatted tables, which were introduced in DB2 Version 9.7. The **tpmon\_client\_wkstn** monitor element is used for snapshot monitors and event monitors that write to tables, files, and pipes.

Table 1994. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |

Table 1995. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Activities  | event_activity        | -              |
| Deadlock    | event_dlconn          | -              |
| Transaction | event_xact            | -              |

## Usage

Use this element to identify the user's machine by node ID, terminal ID, or similar identifiers.

---

## tq\_cur\_send\_spills - Current number of table queue buffers overflowed monitor element

The current number of table queue buffers residing in a temporary table.

Table 1996. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** An agent writing to a table queue may be sending rows to several readers.

The writing agent will overflow buffers to a temporary table when the agent that it is currently sending rows to is not accepting rows and another agent requires rows in order to proceed. Overflowing to temporary table allows both the writer and the other readers to continue processing.

Rows that have been overflowed will be sent to the reading agent when it is ready to accept more rows.

If this number is high, and queries fail with sqlcode -968, and there are messages in db2diag.log indicating that you ran out of temporary space in the TEMP table space, then table queue overflows may be the cause. This could indicate a problem on another node (such as locking). You would investigate by taking snapshots on all the partitions for this query.

There are also cases, perhaps because of the way data is partitioned, where many buffers need to be overflowed for the query. In these cases you will need to add more disk to the temporary table space.

---

## **tq\_id\_waiting\_on - Waited on node on a table queue monitor element**

The identifier of the table queue that is waiting to send or receive data.

*Table 1997. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

### **Usage**

This can be used for troubleshooting.

---

## **tq\_max\_send\_spills - Maximum number of table queue buffers overflows**

Maximum number of table queue buffers overflowed to a temporary table.

### **Element identifier**

tq\_max\_send\_spills

### **Element type**

watermark

*Table 1998. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 1999. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statements | event_subsection      | Always collected |

**Usage** Indicates the maximum number of table queue buffers that have been written to a temporary table.

---

## tq\_node\_waited\_for - Waited for node on a table queue

If the subsection status `ss_status` is *waiting to receive* or *waiting to send* and `tq_wait_for_any` is FALSE, then this is the number of the node that this agent is waiting for.

**Element identifier**

`tq_node_waited_for`

**Element type**

information

*Table 2000. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** This can be used for troubleshooting. You may want to take an application snapshot on the node that the subsection is waiting for. For example, the application could be in a lock wait on that node.

---

## tq\_rows\_read - Number of Rows Read from table queues

Total number of rows read from table queues.

**Element identifier**

`tq_rows_read`

**Element type**

counter

*Table 2001. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 2002. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statements | event_subsection      | Always collected |

**Usage** If monitoring does not indicate that this number is increasing, then processing progress is not taking place.

If there is significant differences in this number between nodes, then some nodes may be over utilized while others are being under utilized.

If this number is large, then there is a lot of data being shipped between nodes, suggest that optimization might improve the access plan.

---

## tq\_rows\_written - Number of rows written to table queues

Total number of rows written to table queues.

**Element identifier**

`tq_rows_written`

**Element type**

counter

Table 2003. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

Table 2004. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statements | event_subsection      | Always collected |

**Usage** If monitoring does not indicate that this number is increasing, then processing progress is not taking place.

If there is significant differences in this number between nodes, then some nodes may be over utilized while others are being under utilized.

If this number is large, then there is a lot of data being shipped between nodes, suggest that optimization might improve the access plan.

---

## mq\_sort\_heap\_rejections - Table queue sort heap rejections monitor element

The number of times that table queues requested for sort heap memory and was rejected due to sort heap threshold being exceeded.

Table 2005. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details                         | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics                                      | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics                     | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS - Get detailed metrics for package cache entries                 | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics         | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                 | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |

Table 2005. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics | REQUEST METRICS BASE             |

Table 2006. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 2007. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2007. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Connection    | event_conn   | -                     |
| Statements    | event_stmt   | -                     |
| Transactions  | event_xact   | -                     |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with the **tq\_sort\_heap\_requests** monitor element to determine if table queues are getting sufficient sort heap memory most of the time. If the ratio of the **tq\_sort\_heap\_rejections** monitor element to the **tq\_sort\_heap\_requests** monitor element is high, database performance may be sub-optimal. Consider increasing the sort heap size.

---

## tq\_sort\_heap\_requests - Table queue sort heap requests monitor element

The number of times that table queues requested for sort heap memory to store data.

Table 2008. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |

Table 2008. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element | Collection Level |
|---|-----------------|------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS | BASE             |

Table 2009. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Connection    | event_conn   | -                     |
| Statements    | event_stmt   | -                     |
| Transactions  | event_xact   | -                     |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

Use this element in conjunction with the **tq\_sort\_heap\_rejections** monitor element to determine if table queues are getting sufficient sort heap memory most of the time. If the ratio of the **tq\_sort\_heap\_rejections** monitor element to the **tq\_sort\_heap\_requests** monitor element is high, database performance may be sub-optimal. Consider increasing the sort heap size.

## tq\_tot\_send\_spills - Total number of table queue buffers overflowed monitor element

Total number of table queue buffers overflowed to a temporary table.

*Table 2010. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details  | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2011. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 2012. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 2012. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch        |
|---------------|--|-----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.             | REQUEST METRICS BASE  |
| Statements    | event_subsection                                     | Always collected      |
| Package cache | Reported in the activity_metrics document.           | ACTIVITY METRICS BASE |

## Usage

Indicates the total number of table queue buffers that have been written to a temporary table. See the **tq\_cur\_send\_spills** monitor element for more information.

---

## tq\_wait\_for\_any - Waiting for any node to send on a table queue

This flag is used to indicate that the subsection is blocked because it is waiting to receive rows from any node.

### Element identifier

tq\_wait\_for\_any

### Element type

information

Table 2013. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** If `ss_status` indicates *waiting to receive data on a table queue* and this flag is TRUE, then the subsection is waiting to receive rows from any node. This generally indicates that the SQL statement has not processed to the point it can pass data to the waiting agent. For example, the writing agent may be performing a sort and will not write rows until the sort has completed. From the `db2expln` output, determine the subsection number associated with the tablequeue that the agent is waiting to receive rows from. You can then examine the status of that subsection by taking a snapshot on each node where it is executing.

---

## ts\_name - Table space being rolled forward monitor element

The name of the table space currently rolled forward.

### Element identifier

ts\_name

### Element type

information

Table 2014. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | rollforward           | Basic          |

**Usage** If a rollforward is in progress, this element identifies the table spaces involved.

---

## txn\_completion\_status - Transaction completion status

This element indicates the status of the transaction.

Table 2015. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | TXNCOMPLETION         | Always collected |

### Usage

For the change history event monitor, the status of the transaction is one of:

- C** Commit
- R** Rollback
- S** Rollback to savepoint

---

## uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed

The number of SQL UPDATE, INSERT, and DELETE statements that were executed.

Table 2016. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 2017. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_db              | Always collected |
| Connection | event_conn            | Always collected |

**Usage** You can use this element to determine the level of database activity at the application or database level.

You can also use the following formula to determine the ratio of UPDATE, INSERT and DELETE statements to the total number of statements:

$$\frac{\text{uid\_sql\_stmts}}{(\text{static\_sql\_stmts} + \text{dynamic\_sql\_stmts})}$$

This information can be useful for analyzing application activity and throughput.

---

## unread\_prefetch\_pages - Unread prefetch pages monitor element

Indicates the number of pages that the prefetcher read in that were never used.

*Table 2018. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Command and Level |
|---|--|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE                     |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE                     |

*Table 2019. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 2020. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch   |
|-------------|-----------------------|------------------|
| Database    | event_db              | Always collected |
| Tablespaces | event_tablespace      | Always collected |
| Connection  | event_conn            | Always collected |

### Usage

If this number is high, prefetchers are causing unnecessary I/O by reading pages into the buffer pool that will not be used.

---

## uow\_comp\_status - Unit of Work Completion Status

The status of the unit of work and how it stopped.

### Element identifier

uow\_comp\_status

### Element type

information

*Table 2021. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl                  | Unit of Work   |
| DCS Application | dcs_appl              | Basic          |

*Table 2022. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Transactions | event_xact            | Always collected |

**Usage** You may use this element to determine if the unit of work ended due to a deadlock or abnormal termination. It may have been:

- Committed due to a commit statement
- Rolled back due to a rollback statement
- Rolled back due to a deadlock
- Rolled back due to an abnormal termination
- Committed at normal application termination.
- Unknown as a result of a FLUSH EVENT MONITOR command for which units of work were in progress.

**Note:** API users should refer to the header file (*sqlmon.h*) containing definitions of database system monitor constants.

---

## uow\_completed\_total - Total completed units of work monitor element

The total number of units of work that completed, either by being committed or rolled back.

*Table 2023. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics                     | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample workload metrics                               | REQUEST METRICS BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                      | REQUEST METRICS BASE             |

*Table 2024. Event Monitoring Information*

| Event Type | Logical Data Grouping                                | Monitor Switch   |
|------------|--|------------------|
| Statistics | event_scstats (reported in the details_xml document) | Always collected |
| Statistics | event_wlstats (reported in the details_xml document) | Always collected |

### Usage

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS function, this monitor element represents the total completed units of work since the last reset of the statistics.

When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS function, this monitor element represents the total completed units of work since the function was executed.

---

## uow\_elapsed\_time - Most Recent Unit of Work Elapsed Time

The elapsed execution time of the most recently completed unit of work.

**Element identifier**

uow\_elapsed\_time

**Element type**

time

*Table 2025. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dcs_appl              | Unit of Work, Timestamp |

**Usage**

Use this element as an indicator of the time it takes for units of work to complete.

This element is composed of two subelements that report time spent as seconds and microseconds (one millionth of a second). The names of the subelements can be derived by adding "\_s" and "\_ms" to the name of this monitor element. To retrieve the total time spent for this monitor element, the values of the two subelements must be added together. For example, if the "\_s" subelement value is 3 and the "\_ms" subelement value is 20, then the total time spent for the monitor element is 3.00002 seconds.

**uow\_id - Unit of work ID monitor element**

The unit of work identifier. The unit of work ID is unique within an application handle.

*Table 2026. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details               | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                        | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics       | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities   | ACTIVITY METRICS BASE            |

*Table 2027. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Locking      | -                     | Always collected |
| Unit of work | -                     | Always collected |
| Activities   | event_activity        | Always collected |

Table 2027. Event Monitoring Information (continued)

| Event Type           | Logical Data Grouping        | Monitor Switch        |
|----------------------|------------------------------|-----------------------|
| Activities           | event_activitystmt           | Always collected      |
| Activities           | event_activityvals           | Always collected      |
| Activities           | event_activitymetrics        | ACTIVITY METRICS BASE |
| Threshold violations | event_thresholdviolations    | Always collected      |
| Change history       | ddlstmtexec<br>txncompletion | Always collected      |

## Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity.

You can also use this element with the **activity\_id** and **appl\_id** monitor elements to uniquely identify an activity.

---

## uow\_lifetime\_avg - Unit of work lifetime average monitor element

The average lifetime of a unit of work. Measured in milliseconds.

Table 2028. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics                        | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample workload metrics                                  | REQUEST METRICS BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service<br>subclasses | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                         | REQUEST METRICS BASE             |

Table 2029. Event Monitoring Information

| Event Type | Logical Data Grouping                                   | Monitor Switch   |
|------------|---|------------------|
| Statistics | event_scstats (reported in the<br>details_xml document) | Always collected |
| Statistics | event_wlstats (reported in the<br>details_xml document) | Always collected |

## Usage

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS function, this monitor element represents the average unit of work lifetime since the last reset of the statistics.

When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS function, this monitor element represents the average unit of work lifetime since the function was executed.

---

## uow\_lock\_wait\_time - Total time unit of work waited on locks monitor element

The total amount of elapsed time this unit of work has spent waiting for locks. The value is given in milliseconds.

### Element identifier

uow\_lock\_wait\_time

### Element type

counter

Table 2030. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Unit of Work   |

**Usage** This element can help you determine the severity of the resource contention problem.

---

## uow\_log\_space\_used - Unit of work log space used monitor element

The amount of log space (in bytes) used in the current unit of work of the monitored application.

Table 2031. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

Table 2032. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Unit of Work   |

Table 2033. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Transactions | event_xact            | Always collected |
| Unit of work | -                     | Always collected |

### Usage

You may use this element to understand the logging requirements at the unit of work level.

---

## uow\_start\_time - Unit of work start timestamp monitor element

The date and time that the unit of work first required database resources.

Table 2034. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document) | ACTIVITY METRICS BASE            |

Table 2035. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dcs_appl              | Unit of Work, Timestamp |

Table 2036. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Unit of work | -                     | -              |
| Transaction  | event_xact            | -              |

### Usage

This resource requirement occurs at the first SQL statement execution of that unit of work:

- For the first unit of work, it is the time of the first database request (SQL statement execution) after **conn\_complete\_time**.
- For subsequent units of work, it is the time of the first database request (SQL statement execution) after the previous COMMIT or ROLLBACK.

**Note:** The *SQL Reference* defines the boundaries of a unit of work as the COMMIT or ROLLBACK points.

The database system monitor excludes the time spent between the COMMIT/ROLLBACK and the next SQL statement from its definition of a unit of work. This measurement method reflects the time spent by the database manager in processing database requests, separate from time spent in application logic before the first SQL statement of that unit of work. The unit of work elapsed time does include the time spent running application logic between SQL statements within the unit of work.

You may use this element with the **uow\_stop\_time** monitor element to calculate the total elapsed time of the unit of work and with the **prev\_uow\_stop\_time** monitor element to calculate the time spent in the application between units of work.

You can use the **uow\_stop\_time** and the **prev\_uow\_stop\_time** monitor elements to calculate the elapsed time for the *SQL Reference* definition of a unit of work.

---

## uow\_status - Unit of Work Status

The status of the unit of work.

**Element identifier**  
uow\_status

**Element type**  
information

*Table 2037. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Transactions | event_xact            | Always collected |

**Usage** You may use this element to determine the status of a unit of work. API users should refer to the sqlmon.h header file containing definitions of database system monitor constants.

---

## uow\_stop\_time - Unit of work stop timestamp monitor element

The date and time that the most recent unit of work completed, which occurs when database changes are committed or rolled back.

*Table 2038. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dcs_appl              | Unit of Work, Timestamp |

*Table 2039. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Unit of work | -                     | -              |

### Usage

Use this element with the **prev\_uow\_stop\_time** monitor element to calculate the total elapsed time between COMMIT/ROLLBACK points, and with the **uow\_start\_time** monitor element to calculate the elapsed time of the latest unit of work.

The timestamp contents will be set as follows:

- When the application has completed a unit of work and has not yet started a new one (as defined in the **uow\_start\_time** monitor element), this element reports a valid, non-zero timestamp.
- When the application is currently executing a unit of work, this element reports zeros.
- When the application first connects to the database, this element is set to the value of the **conn\_complete\_time** monitor element

As a new unit of work is started, the contents of this element are moved to the **prev\_uow\_stop\_time** monitor element.

---

## uow\_throughput - Unit of work throughput monitor element

The completion rate of units of work measured in units of work per second.

*Table 2040. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_SAMPLE_SERVICE_CLASS_METRICS<br>- Get sample service class metrics                        | REQUEST METRICS BASE             |
| MON_SAMPLE_WORKLOAD_METRICS -<br>Get sample workload metrics                                  | REQUEST METRICS BASE             |
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service<br>subclasses | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                         | REQUEST METRICS BASE             |

*Table 2041. Event Monitoring Information*

| Event Type | Logical Data Grouping                                   | Monitor Switch   |
|------------|---|------------------|
| Statistics | event_scstats (reported in the<br>details_xml document) | Always collected |
| Statistics | event_wlstats (reported in the<br>details_xml document) | Always collected |

### Usage

When returned by the WLM\_GET\_SERVICE\_SUBCLASS\_STATS or the WLM\_GET\_WORKLOAD\_STATS function, this monitor element represents the unit of work throughput since the last reset of the statistics.

When returned by the MON\_SAMPLE\_SERVICE\_CLASS\_METRICS or the MON\_SAMPLE\_WORKLOAD\_METRICS function, this monitor element represents the unit of work throughput since the function was executed.

---

## uow\_total\_time\_top - UOW total time top monitor element

High watermark for unit of work lifetime, in milliseconds.

*Table 2042. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service<br>subclasses | REQUEST METRICS BASE             |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                         | REQUEST METRICS BASE             |

*Table 2043. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_wlstats         | Always collected |
| Statistics | event_scstats         | Always collected |

## Usage

This element can be used to help determine whether or not the UOWTOTALTIME threshold is effective and can also help to determine how to configure such a threshold.

For service classes, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE.

For workloads, this monitor element returns -1 if COLLECT AGGREGATE ACTIVITY DATA for the workload is set to NONE.

For a service class, measurements taken for this high watermark are computed for the service class assigned by the workload. Any mapping by a work action set to change the service class of an activity does not affect this high watermark.

---

## update\_sql\_stmts - Updates

This element contains a count of the total number of times the federated server has issued an UPDATE statement to this data source on behalf of any application from the start of the federated server instance, or the last reset of the database monitor counters.

*Table 2044. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |
| Application    | appl_remote           | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** Use this element to determine the level of database activity directed against this data source by the federated server or an application.

You can also use this element to determine the percentage of write activity against this data source by the federated server or an application, with the following formula:

$$\text{write\_activity} = \frac{(\text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}{(\text{SELECT statements} + \text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}$$

---

## update\_time - Update Response Time

This element contains the aggregate amount of time, in milliseconds, that it has taken this data source to respond to UPDATES from all applications or a single application running on this federated server instance from the start of the federated server instance, or the last reset of the database monitor counters.

*Table 2045. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Timestamp      |
| Application    | appl_remote           | Timestamp      |

For snapshot monitoring, this counter can be reset.

The response time is measured as the difference in time between the time the federated server submits an UPDATE statement to the data source, and the time the data source responds to the federated server, indicating the UPDATE has been processed.

**Usage** Use this element to determine how much actual time transpires while waiting for UPDATES to this data source to be processed. This information can be useful for capacity planning and tuning.

---

## usage\_list\_last\_state\_change - Last state change monitor element

A timestamp that indicates when the value of the **usage\_list\_state** monitor element last changed.

*Table 2046. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## usage\_list\_last\_updated - Usage list last updated monitor element

A timestamp indicating the last time that a particular section was updated. The section is represented by the values of the **executable\_id** and **mon\_interval\_id** monitor elements.

*Table 2047. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list  | ACTIVITY METRICS BASE            |

---

## usage\_list\_mem\_size - Usage list memory size monitor element

The total amount of memory that is allocated for a particular usage list, in kilobytes.

*Table 2048. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## usage\_list\_name - Usage list name monitor element

A usage list name.

*Table 2049. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list  | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list             | ACTIVITY METRICS BASE            |

---

## usage\_list\_num\_references - Number of references monitor element

The total number of times that a specific section referenced a specific object since the section was added to the usage list.

*Table 2050. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list  | ACTIVITY METRICS BASE            |

---

## usage\_list\_num\_ref\_with\_metrics - Number of references with metrics monitor element

The total number of times that a specific section referenced a specific object since the section was added to the usage list and had statistics collected.

*Table 2051. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table<br>usage list  | ACTIVITY METRICS BASE            |

---

## usage\_list\_schema - Usage list schema monitor element

The name of the schema of a usage list.

*Table 2052. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index usage list | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table<br>function - Returns information from a table usage list  | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list          | ACTIVITY METRICS BASE            |

---

## usage\_list\_size - Usage list size monitor element

The maximum number of entries that a particular usage list can hold.

*Table 2053. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## usage\_list\_state - Usage list state monitor element

The state of a particular usage list.

Possible values are as follows:

- A** Active.
- F** Failed to activate.
- I** Inactive.
- P** Activation pending.

*Table 2054. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## usage\_list\_used\_entries - Usage list used entries monitor element

The number of entries currently in a usage list. If the usage list is in an Inactive state, this monitor element represents the number of entries that were in this usage list when it was last active for monitoring.

*Table 2055. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## usage\_list\_wrapped - Usage list wrap indicator monitor element

An indicator of whether a particular usage list has wrapped. When a usage list becomes full, the default behavior is for entries to wrap, which means the oldest entries get replaced the by newest entries.

Possible values are Y and N.

Table 2056. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_USAGE_LIST_STATUS table<br>function - Returns the status of a usage list | ACTIVITY METRICS BASE            |

---

## user\_cpu\_time - User CPU time monitor element

The total *user* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

When either the statement monitor switch or the timestamp switch is not turned on, this element is not collected and -1 is written instead.

Table 2057. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Connection   | event_conn            | Always collected |
| Transactions | event_xact            | Always collected |
| Statements   | event_stmt            | Always collected |
| Activities   | event_activity        | Always collected |

### Usage

This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

**Note:** If this information is not available for your operating system, this element will be set to 0.

**Note:** Due to the differences in granularity with which the DB2 system collects statistics, the value of the **total\_exec\_time** monitor element might not equal the sum of values of **system\_cpu\_time** and **user\_cpu\_time** monitor elements. In this case, the sum of **system\_cpu\_time** and **user\_cpu\_time** monitor elements more accurately reflects the actual total execution time.

---

## utility\_dbname - Database Operated on by Utility

The database operated on by the utility.

Table 2058. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

---

## utility\_description - Utility Description

A brief description of the work a utility is performing. For example, a rebalance invocation may contain "Tablespace ID: 2" representing that this rebalancer is working on table space with ID 2. The format of this field is dependent on the class of utility and is subject to change between releases.

*Table 2059. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

---

## utility\_detail - Utility detail

This element contains a brief description of the work a utility is performing.

*Table 2060. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTART             | Always collected |

### Usage

A brief description of the work a utility is performing, including some options specified for the utility. For example, a record for the invocation of REORG includes a partially reconstructed command string including some of the different options used by the utility such as access mode . The format of this field is dependent on the type of utility and might change between releases.

---

## utility\_id - Utility ID

The unique identifier corresponding to the utility invocation.

*Table 2061. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Command and Level |
|--|--|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE                        |
| WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES table function - Return a list of activities                       | ACTIVITY METRICS BASE                        |

*Table 2062. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

---

## utility\_invocation\_id - Utility invocation ID

A unique identifier corresponding to the utility invocation.

Table 2063. Event Monitoring Information

| Event Type     | Logical Data Grouping   | Monitor Element Collection Level |
|----------------|---|----------------------------------|
| Change History | changesummary<br>utillocation<br>utilphase<br>utilstart<br>utilstop | Always collected                 |

### Usage

The **utility\_invocation\_id** is a binary token that uniquely identifies a given invocation of a utility. The **utility\_invocation\_id** is the same on each member where the utility is executing. The **utility\_invocation\_id** will retain its uniqueness across database deactivation, reactivation, and member shutdown, allowing quick identification of all event monitor records corresponding to a given invocation of a utility.

---

## utility\_invoker\_type - Utility Invoker Type

This element describes how a utility was invoked.

Table 2064. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

Table 2065. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change history | utilstart             | Always collected |

### Usage

Use this element to determine how a utility was invoked. For example, you can use it to determine whether a utility was invoked automatically by DB2 or by a user. The values for this element, listed as follows, are defined in sqlmon.h.

| API Constant              | Utility                                  |
|---------------------------|--|
| SQLM_UTILITY_INVOKER_USER | Utility was invoked by user              |
| SQLM_UTILITY_INVOKER_AUTO | Utility was invoked automatically by DB2 |

For the change event history monitor, this element indicates how a utility was invoked:

**USER** Utility was invoked by a user

**AUTO** Utility was invoked automatically by DB2

---

## utility\_operation\_type - Utility operation type

Indicates the type of utility operation.

*Table 2066. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTART             | Always collected |

### Usage

For the change history event monitor, this element contains details about utility events (UTILITY\_TYPE) that were started .

If UTILITY\_TYPE is BACKUP, one of:

- D     Delta
- I     Incremental
- F     Full

If UTILITY\_TYPE is LOAD, one of:

- I     Insert
- R     Replace
- S     Restart
- T     Terminate

If UTILITY\_TYPE is MOVETABLE, one of:

- A     Cancel
- C     Copy
- I     Init
- L     Cleanup
- M     Move
- R     Replay
- S     Swap
- V     Verify

If UTILITY\_TYPE is REDISTRIBUTE, one of:

- A     Abort
- C     Continue
- D     Default
- T     Target Map

If UTILITY\_TYPE is REORG, one of:

- A     Reorganize all table indexes
- I     Index reorganization
- N     Inplace table reorganization

- R Reorganize table reclaim extents
- T Classic table reorganization

If UTILITY\_TYPE is RESTORE, one of:

- A Incremental automatic
- B Incremental abort
- F Full
- M Incremental manual

If UTILITY\_TYPE is ROLLFORWARD, one of:

- E End of logs
- P Point in time

If UTILITY\_TYPE is RUNSTATS, one of:

- A All indexes on a table
- I Index
- T Table

---

## utility\_phase\_detail - Utility phase detail

This element reserved for future use.

*Table 2067. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILPHASE             | Always collected |

---

## utility\_phase\_type - Utility phase type

Identifies the utility phase type.

*Table 2068. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILPHASE             | Always collected |

### Usage

For the change history event monitor, if the utility\_type element is BACKUP, the phase type is:

#### BACKUPTS

Backup table space

---

## utility\_priority - Utility Priority

Utility priority specifies the amount of relative importance of a throttled utility with respect to its throttled peers. A priority of 0 implies that a utility is executing unthrottled. Non-zero priorities must fall in the range of 1-100, with 100 representing the highest priority and 1 representing the lowest.

Table 2069. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

Table 2070. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change history | utilstart             | Always collected |

---

## utility\_start\_time - Utility Start Time

The date and time when the current utility was originally invoked.

Table 2071. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

---

## utility\_start\_type - Utility start type

This element indicates how a utility was started.

Table 2072. Event Monitoring Information

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTART             | Always collected |

### Usage

For the change history event monitor, utility start information is one of:

- RESUME
- START

---

## utility\_state - Utility State

This element describes the state of a utility.

**Element identifier**  
utility\_state

**Element type**  
information

Table 2073. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

**Usage** Use this element to determine the state of an active utility. The values for this field, listed as follows, are defined in sqlmon.h.

| API Constant               | Description          |
|----------------------------|----------------------|
| SQLM_UTILITY_STATE_EXECUTE | Utility is executing |

| API Constant             | Description   |
|--------------------------|---|
| SQLM_UTILITY_STATE_WAIT  | Utility is waiting for an event to occur before resuming progress |
| SQLM_UTILITY_STATE_ERROR | Utility has encountered an error                                  |

---

## utility\_stop\_type - Utility stop type

This element indicates how a utility was stopped.

*Table 2074. Event Monitoring Information*

| Event Type     | Logical Data Grouping | Monitor Switch   |
|----------------|-----------------------|------------------|
| Change History | UTILSTOP              | Always collected |

### Usage

For the change history event monitor, a utility was stopped in one of the following ways:

- PAUSE
- STOP

---

## valid - Section validity indicator monitor element

Indicates whether the dynamic SQL statement section is valid. For static SQL statements, the value of this monitor element is always Y.

*Table 2075. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries  | ACTIVITY METRICS BASE            |

### Usage

Valid values for this monitor element are Y and N. An invalid section will be implicitly prepared by the system when next used.

---

## utility\_type - Utility Type

The class of utility.

*Table 2076. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

Table 2077. Event Monitoring Information

| Event Type     | Logical Data Grouping   | Monitor element collection level |
|----------------|---|----------------------------------|
| Change history | changesummary<br>utillocation<br>utilphase<br>utilstart<br>utilstop | Always collected                 |

## Usage

The values for this element can be any of the constants defined in `sqlmon.h` with names beginning "SQLM\_UTILITY\_".

For the change history event monitor, the utility type is one of:

- BACKUP
- LOAD
- MOVETABLE
- REDISTRIBUTE
- REORG
- RESTORE
- ROLLFORWARD
- RUNSTATS

---

## valid - Section validity indicator monitor element

Indicates whether the dynamic SQL statement section is valid. For static SQL statements, the value of this monitor element is always Y.

Table 2078. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries  | ACTIVITY METRICS BASE            |

## Usage

Valid values for this monitor element are Y and N. An invalid section will be implicitly prepared by the system when next used.

---

## vectored\_ios - Number of vectored I/O requests monitor element

The number of vectored I/O requests. More specifically, the number of times DB2 performs prefetching of pages into the page area of the buffer pool.

*Table 2079. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics          | DATA OBJECT METRICS BASE         |
| MON_GET_CONTAINER table function -<br>Get table space container metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics          | DATA OBJECT METRICS BASE         |

*Table 2080. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

### Usage

Use this element to determine how often vectored I/O is being done. The number of vectored I/O requests is monitored only during prefetching.

---

## version - Version of Monitor Data

The version of the database manager that produced the event monitor data stream.

*Table 2081. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

### Usage

The data structures used by the event monitor may change between releases of the database manager. As a result, your monitor applications should check the version of the data stream to determine if they can process the data they will be receiving.

For this release, this element is set to the API constant `SQLM_DBMON_VERSION9_5`.

---

## virtual\_mem\_free - Free virtual memory monitor element

The amount of virtual memory available on this host that is not allocated to any process, in MB.

*Table 2082. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## virtual\_mem\_reserved - Reserved virtual memory monitor element

The amount of virtual memory reserved by running processes, in MB.

*Table 2083. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## virtual\_mem\_total - Total virtual memory monitor element

The total amount of virtual memory available on this host, in MB.

*Table 2084. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ENV_GET_SYSTEM_RESOURCES table<br>function - Return system information | ACTIVITY METRICS BASE            |

---

## wl\_work\_action\_set\_id - Workload work action set identifier monitor element

If this activity has been categorized into a work class of workload scope, this monitor element shows the ID of the work action set associated with the work class set to which the work class belongs. Otherwise, this monitor element shows the value of 0.

*Table 2085. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table<br>function - Get complete activity details<br>(reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 2086. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this monitor element, together with the `wl_work_class_id` monitor element, to uniquely identify the workload work class of the activity, if one exists.

---

## wl\_work\_class\_id - Workload work class identifier monitor element

If this activity has been categorized into a work class of workload scope, then this monitor element displays the identifier of the work class. Otherwise, this monitor element displays the value of 0.

*Table 2087. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE            |

*Table 2088. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Activities | event_activity        | Always collected |

### Usage

Use this monitor element, together with the **wl\_work\_action\_set\_id** monitor element, to uniquely identify the workload work class of the activity, if one exists.

---

## wlm\_queue\_assignments\_total - Workload manager total queue assignments monitor element

The number of times that activities have been queued by a WLM threshold.

*Table 2089. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 2089. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 2090. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

## wlm\_queue\_time\_total - Workload manager total queue time monitor element

The time spent waiting on a WLM queuing threshold. This value is given in milliseconds.

Table 2091. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                           | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |

Table 2091. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE             |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 2092. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml) document) | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |

---

## wlo\_completed\_total - Workload occurrences completed total monitor element

The number of workload occurrences to complete since last reset.

*Table 2093. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics | ACTIVITY METRICS BASE            |

*Table 2094. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |

### Usage

Use this element to determine how many occurrences of a given workload are driving work into the system.

---

## work\_action\_set\_id - Work action set ID monitor element

The ID of the work action set to which this statistics record applies.

*Table 2095. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_histogrambin    | Always collected |
| Statistics | event_wcstats         | Always collected |

### Usage

Use this element in conjunction with other activity history elements for analysis of the behavior of an activity or with other statistics elements for analysis of a work class.

The value of this element is 0 when the following conditions are met:

- The element is reported in an event\_histogrambin logical data group.
- The histogram data is collected for an object that is not a work class.

---

## work\_action\_set\_name - Work action set name monitor element

The name of the work action set to which the statistics shown as part of this event are associated.

*Table 2096. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics           | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS table function - Return work action set statistics | ACTIVITY METRICS BASE            |

Table 2097. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this element along with the **work\_class\_name** element to uniquely identify the work class whose statistics are being shown in this record or to uniquely identify the work class which is the domain of the threshold queue whose statistics are shown in this record.

---

## work\_class\_id - Work class ID monitor element

The identifier of the work class to which this statistics record applies.

Table 2098. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_wcstats         | Always collected |
| Statistics | event_histogrambin    | Always collected |

## Usage

Use this element in conjunction with other statistics elements for analysis of a work class.

The value of this element is 0 when the following conditions are met:

- The element is reported in an event\_histogrambin logical data group.
- The histogram data is collected for an object that is not a work class.

---

## work\_class\_name - Work class name monitor element

The name of the work class to which the statistics shown as part of this event are associated.

Table 2099. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics           | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS table function - Return work action set statistics | ACTIVITY METRICS BASE            |

Table 2100. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this element along with the `work_action_set_name` element to uniquely identify the work class whose statistics are being shown in this record or to uniquely identify the work class which is the domain of the threshold queue whose statistics are shown in this record.

---

## workload\_id - Workload ID monitor element

An integer that uniquely identifies a workload.

*Table 2101. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_WORKLOAD table function - Get workload metrics                  | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample                                | ACTIVITY METRICS BASE            |

*Table 2102. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |

*Table 2103. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Locking      | -   | Always collected      |
| Unit of work | -   | Always collected      |
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | Always collected      |
| Statistics   | event_wlstats   | Always collected      |
| Statistics   | event_histogrambin                                    | Always collected      |
| Activities   | event_activity  | Always collected      |

## Usage

Use this ID to uniquely identify the workload to which this activity, application, histogram bin, or workload statistics record belongs.

The value of this element is 0 when the following conditions are met:

- The element is reported in an event\_histogrambin logical data group.
- The histogram data is collected for an object that is not a workload.

---

## workload\_name - Workload name monitor element

Name of the workload.

*Table 2104. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                         | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics        | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD table function - Get workload metrics                                 | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics                | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample   | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics                 | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences  | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics                     | ACTIVITY METRICS BASE            |

*Table 2105. Event Monitoring Information*

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Locking      | -   | Always collected      |
| Unit of work | -   | Always collected      |
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | Reported in the system_metrics document.              | Always collected      |
| Statistics   | event_wlstats   | Always collected      |

### Usage

In the statistics event monitor and workload table functions, the workload name identifies the workload for which statistics or metrics are being collected and

reported. In the unit of work event monitor and unit of work table functions, the workload name identifies the workload that the unit of work was associated with.

Use the workload name to identify units of work or sets of information that apply to a particular workload of interest.

---

## workload\_occurrence\_id - Workload occurrence identifier monitor element

The ID of the workload occurrence to which this activity belongs.

*Table 2106. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics                            | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics        | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table<br>function - list agents running in a service<br>class   | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list<br>workload occurrences | ACTIVITY METRICS BASE            |

*Table 2107. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Unit of work | -                     | Always collected |
| Activities   | event_activity        | Always collected |

### Usage

Use this to identify the workload occurrence that submitted the activity.

---

## workload\_occurrence\_state - Workload occurrence state monitor element

The state of the workload occurrence.

*Table 2108. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document) | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD<br>_OCCURRENCES table function - list<br>workload occurrences                                | ACTIVITY METRICS BASE            |

## Usage

Possible values include:

### DECOUPLED

Workload occurrence does not have a coordinator agent assigned (concentrator case).

### DISCONNECTPEND

Workload occurrence is disconnecting from the database.

### FORCED

Workload occurrence has been forced.

### INTERRUPTED

Workload occurrence has been interrupted.

### QUEUED

Workload occurrence coordinator agent is queued by a workload management queuing threshold. In a partitioned database environment, this state may indicate that the coordinator agent has made an RPC another member to obtain threshold tickets and has not yet received a response.

### TRANSIENT

Workload occurrence has not yet been mapped to a service superclass.

### UOWEXEC

Workload occurrence is processing a request.

### UOWWAIT

Workload occurrence is waiting for a request from the client.

---

## x\_lock\_escals - Exclusive lock escalations monitor element

The number of times that locks have been escalated from several row locks to one exclusive table lock, or the number of times an exclusive lock on a row caused the table lock to become an exclusive lock.

*Table 2109. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 2110. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch   |
|--------------|-----------------------|------------------|
| Database     | event_db              | Always collected |
| Connection   | event_conn            | Always collected |
| Transactions | event_xact            | Always collected |

## Usage

Other applications cannot access data held by an exclusive lock; therefore it is important to track exclusive locks since they can impact the concurrency of your data.

A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application. The amount of lock list space available is determined by the **locklist** and **maxlocks** configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs.

See the **lock\_escals** monitor element for possible causes and resolutions to excessive exclusive lock escalations.

An application may be using exclusive locks when share locks are sufficient. Although share locks may not reduce the total number of lock escalations share lock escalations may be preferable to exclusive lock escalations.

## xda\_object\_pages - XDA Object Pages

The number of disk pages consumed by XML storage object (XDA) data.

**Element identifier**

xda\_object\_pages

**Element type**

information

*Table 2111. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 2112. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Tables     | event_table           | Always collected |

**Usage** This element provides a mechanism for viewing the actual amount of space consumed by XML storage object (XDA) data in a particular table. This element can be used in conjunction with a table event monitor to track the rate of XML storage object data growth over time.

## xda\_object\_l\_pages - XML storage object (XDA) data logical pages monitor element

The number of logical pages used on disk by XML storage object (XDA) data.

*Table 2113. Table Function Monitoring Information*

| Table Function                                   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLE table function - Get table metrics | ACTIVITY METRICS BASE            |

**Usage**

- This value might be less than the amount of space physically allocated for the object. This can happen when you use the REUSE STORAGE option of the TRUNCATE statement. This option causes storage allocated for the table to

continue to be allocated, although the storage will be considered empty. In addition, the value for this monitor element might be less than the amount of space logically allocated for the object, because the total space logically allocated includes a small amount of additional meta data.

To retrieve an accurate measure of the logical or physical size of an object, use the ADMIN\_GET\_TAB\_INFO\_V97 function. This function provides more accurate information about the size of objects than you can obtain by multiplying the number of pages reported for this monitor element by the page size.

## xid - Transaction ID

A unique transaction identifier (across all databases) generated by a transaction manager in a two-phase commit transaction.

*Table 2114. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_appl              | Unit of Work   |

**Usage** This identifier can be used to correlate the transaction generated by the transaction manager with the transactions executed against multiple databases. It can be used to help diagnose transaction manager problems by tying database transactions involving a two-phase commit protocol with the transactions originated by the transaction manager.

## xmld - XML ID monitor element

A unique document ID. The ID is derived as follows:

<event\_header>\_<event\_id>\_<event\_type>\_<event\_timestamp>\_<partition>.

*Table 2115. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| EVMON_FORMAT_UE_TO_TABLES procedure - move an XML document to relational tables | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| EVMON_FORMAT_UE_TO_XML table function - convert unformatted events to XML       | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |

*Table 2116. Event Monitoring Information*

| Event Type    | Logical Data Grouping | Monitor Switch |
|---------------|-----------------------|----------------|
| Locking       |                       |                |
| Package Cache |                       |                |

## xquery\_stmts - XQuery Statements Attempted

The number of XQuery statements executed for an application or database.

**Element identifier**

xquery\_stmts

**Element type**

counter

*Table 2117. Snapshot Monitoring Information*

| <b>Snapshot Level</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-----------------------|------------------------------|-----------------------|
| Database              | dbase                        | Basic                 |
| Application           | appl                         | Basic                 |

For snapshot monitoring, this counter can be reset.

*Table 2118. Event Monitoring Information*

| <b>Event Type</b> | <b>Logical Data Grouping</b> | <b>Monitor Switch</b> |
|-------------------|------------------------------|-----------------------|
| Database          | event_db                     | Always collected      |
| Connection        | event_conn                   | Always collected      |

**Usage** You can use this element to gauge the activity of native XQuery language requests. This does not include embedded XQuery language requests such as `xmlquery`, `xmltable`, or `xmlexist`.

---

## Part 3. Monitoring in a DB2 pureScale environment

The IBM DB2 pureScale Feature provides a robust, highly available database processing environment. Problems that might arise with the operation of one or more host systems in the DB2 pureScale instance can typically be addressed without interrupting access to data. Ironically, this characteristic of high-availability in a DB2 pureScale environment can mask issues that might lead to less than optimal performance. Monitoring certain aspects of your DB2 pureScale environment can help you recognize and address such issues.

For example, you might have a hardware problem that causes a member or cluster caching facility (also known as a CF) to fail over repeatedly to another host. However, because recovery is in most cases automatic, you might never be aware of the problem. If the problem goes undetected and uncorrected, you will not be realizing the full performance potential of your DB2 pureScale environment .

For this reason, doing some level of ongoing, operational monitoring of your DB2 pureScale instance is recommended. It can help you answer such questions as:

- Are all components of my DB2 pureScale instance running?
- If a member or CF has failed, was it able to restart successfully?
- Is my CF running on its preferred primary host? Or has it failed over to another host?
- Is another CF in a state where it is ready to take over if the primary CF fails?

A good place to start when you want to see the overall status of your DB2 pureScale environment is to examine the operational status of the hosts, members and CFs in your instance. By examining state and alert information reported for each of these entities, you can form an overall picture of how well your DB2 pureScale instance is functioning.

Beyond looking at the overall state of your DB2 pureScale instance, you can also use the DB2 monitoring infrastructure to examine monitor elements that provide information about specific aspects of the DB2 pureScale instance. This information can help you to better understand where configuration and application design issues might detract from overall system performance.



---

## Chapter 12. Status monitoring of a DB2 pureScale instance

Viewing the overall status of the components in a DB2 pureScale instance can give you a picture of how effectively it is functioning. You can view this information using table functions and administrative views. There are also DB2 command-line processor (CLP) commands and system commands that you can use to display operational status for your instance.

The information returned from these table functions, views, and commands provides you with a high-level view of the state of your DB2 pureScale instance. For example, you can use these interfaces to find answer to questions like the ones that follow: :

- Are the hosts that comprise a DB2 pureScale instance active or not?
- Which hosts are functioning as cluster caching facilities or member servers?
- Are there hosts that are functioning as both a member and a cluster caching facility?
- Are there hosts that are running more than one member?
- Is a member running on its preferred home host?
- For configurations with more than one cluster caching facility, which one is functioning as the primary server? What is the current state of the other, non-primary cluster caching facilities, PEER, or CATCHUP?
- What is the current *state* of the members that comprise an instance? For example, has any member been stopped using the **db2stop** command? Or is any member currently waiting to fail back to its home host?
- Do any cluster caching facility or member servers indicate an active *alert* that requires investigation?

Included in the information returned by many of these interfaces is *state* and *alert* information. The state of a host, a member or a cluster caching facility reflects the current operational capacity of the object in question. For example, a member can be STARTED, STOPPED, RESTARTING, WAITING\_FOR\_FAILBACK, ERROR or UNKNOWN.

Alerts are used to indicate that there is something that requires further investigation. If an alert is raised for one or more cluster caching facilities or members in an instance, problems might exist that require investigation.

---

### Interfaces for retrieving status information for DB2 pureScale instances

To see the overall status of various components in your DB2 pureScale instance, you can choose from a number of administrative views, table functions, and CLP commands. In addition, there are commands that you can use from the system command prompt or shell, which are useful when any of the server components are not running.

You can view state and alert information about your DB2 pureScale instance in the following ways:

- “Table functions and administrative views” on page 1436
- “CLP commands” on page 1436
- “System prompt (shell) commands” on page 1437.

## Table functions and administrative views

Table functions provide a flexible interface for retrieving information about your system. Most table functions accept parameters to narrow the scope of the information returned. For example, you might be interested in viewing information about a specific host.

Administrative views provide quick and easy access to system information. Unlike table functions, you cannot pass parameters to an administrative view to narrow the scope of a query. Generally, information for all objects relevant to the view (for example, hosts, cluster caching facilities (also known as CFs) or members) in the system are returned. You can always filter the output of administrative views using SQL, however.

The following table functions and their corresponding administrative views return information about the overall status of a DB2 pureScale instance:

*Table 2119. Table functions and administrative views that display status information for components in a DB2 pureScale instance*

| Interface  | Description  |
|--|--|
| DB2_GET_CLUSTER_HOST_STATE table function<br>DB2_CLUSTER_HOST_STATE administrative view              | These interfaces provide basic information about the hosts that comprise a DB2 pureScale instance. They return a list of hosts and associated state information.   |
| DB2_GET_INSTANCE_INFO table function<br>DB2_MEMBER administrative view<br>DB2_CF administrative view | These interfaces provide more detailed information about a DB2 pureScale instance. They return information about the role in the instance each host plays (cluster caching facility or member), whether each cluster caching facility or member is running on its home host, and connection information for each host. |
| DB2_INSTANCE_ALERTS administrative view  | This interface provides information about alerts in a DB2 pureScale instance.  |

**Note:** Each of the preceding interfaces can be used in both DB2 pureScale instances and other DB2 instances. The results they return for each might be different. For example, the DB2\_MEMBER administrative view can be used for both types of instances; however, in an instance outside of a DB2 pureScale environment, no state or alert information is included in the information returned. You must remember to interpret the results returned from the functions and views in the context of the type of instance that is being queried. Consult the reference topics for each of the specific table functions or administrative views for details.

## CLP commands

The following commands can be used from the DB2 command-line processor (CLP):

Table 2120. CLP commands that display status information for components in a DB2 pureScale instance

| CLP command                      | Description   |
|----------------------------------|---|
| <b>LIST INSTANCE</b>             | This command returns information about the state of members, hosts, and cluster caching facilities.   |
| <b>LIST INSTANCE SHOW DETAIL</b> | This command is an extension of the LIST INSTANCE command that returns added information, including partition number and connection information for members, hosts, and cluster caching facilities. |

## System prompt (shell) commands

The following commands can be used from the system or shell prompt:

Table 2121. System prompt (shell) commands that display information for components in a DB2 pureScale instance

| Command                         | Description   |
|---------------------------------|---|
| <b>db2instance -list</b>        | This command returns status information about members, hosts, and cluster caching facilities in a DB2 pureScale instance. It can be used even if there is no current database connection, or if the instance is stopped. In the latter case, the <b>db2instance -list</b> command works with the cluster manager to report the information about the hosts in the DB2 pureScale instance. There are several options available that restrict the output to members only, or cluster caching facilities only. |
| <b>db2cluster -list options</b> | This command can be used to view information about DB2 pureScale instances. There are several additional options to choose from for this command; when you specify the <b>-list</b> option, you must also specify additional options to specify what to include in the command output.  |

---

## Values for member and cluster caching facility states and alerts

Many of the table functions, administrative views, and commands that you can use to query the status of components in a DB2 pureScale environment return *state* and *alert* information. The *state* of a host, member or cluster caching facility (also known as a CF) reflects its operational status. An *alert* for a host, member or CF is an indication that a problem exists that might require investigation or intervention.

### States for hosts, members and cluster caching facilities

State information is returned by many of the table functions, administrative views, and commands that you can use to query the status of the components of a DB2 pureScale environment. The possible values for the state of each component are shown in Table 2122 on page 1438:

Table 2122. Possible states for hosts, members and cluster caching facilities

| Component                     | Possible states       | Description  |
|-------------------------------|-----------------------|--|
| Host                          | ACTIVE                | Host is available for use. This means that the host system is running and can respond to operating system or networking commands, such as the TCP/IP <b>ping</b> command.  |
|                               | INACTIVE              | Host is not available for use. This means that the host system is not running, not available or not responding to system commands. The reasons for being in this state can range from a power loss at the host to connection or networking issues.   |
| cluster caching facility (CF) | STOPPED               | CF has been manually stopped using the <b>db2stop</b> command as part of a normal shutdown by the administrator.   |
|                               | RESTARTING            | CF is in the process of starting, either from the <b>db2start</b> command, or after a CF failure.  |
|                               | BECOMING_PRIMARY      | Once a CF has started, it attempts to take on the role of the primary CF in the instance if no other CF already has this role.   |
|                               | PRIMARY               | The CF is operating normally as the primary CF.  |
|                               | CATCHUP ( <i>n%</i> ) | When a backup CF is initially started, it does not contain any information from the primary CF. During CATCHUP state, the backup CF is in the process of obtaining a copy of all relevant information from the primary CF. This information that enables it to assume the role of primary CF if the primary CF fails. <i>n%</i> indicates how far along the backup CF is in the process of copying information from the primary CF. When this copying process is complete, the backup CF moves into PEER state.<br><b>Note:</b> When you view the status of the non-primary CF using the command <b>db2instance -list</b> , it will be in CATCHUP state until a connection is made to the database. Once the first connection is made, the process of copying data from the primary CF begins. |
|                               | PEER                  | The backup CF is ready to take over the responsibilities of the primary CF in the event of a primary CF failure. Duplexing continues while the backup CF is in PEER state.   |
|                               | ERROR                 | DB2 cluster services could not automatically restart the CF. When the CF reflects an ERROR state, the ALERT field is always set to YES, indicating intervention and investigation is required by the administrator. DB2 cluster services no longer attempts to restart the CF once it is in the ERROR state unless the alert has been cleared.<br><br>The ERROR state can also occur if a connection to a CF cannot be established to query its state. In this case, the ALERT field is not set to YES because the problem might be temporary.   |

Table 2122. Possible states for hosts, members and cluster caching facilities (continued)

| Component | Possible states      | Description   |
|-----------|----------------------|---|
| Member    | STARTED              | Member is started in the instance and operating normally. All databases are consistent and member is ready to accept or is already accepting connections to databases. If a member failed and started again, it is possible that the process model has started, but that crash recovery of the database is not yet complete. Use the <b>LIST UTILITIES SHOW DETAIL</b> command from any member to monitor the recovery progress.  |
|           | STOPPED              | Member has been manually stopped using the <b>db2stop</b> command as part of a normal shutdown by the administrator.  |
|           | RESTARTING           | Member is in the process of starting or restarting. If the current host is the same as the home host, then a local member restart is taking place. If current host is different from the home host, then a member has failed over to the current host, and is restarting in light mode.   |
|           | WAITING_FOR_FAILBACK | The process model for this member has successfully restarted on the current host in light mode. The member is waiting for its home host to become available, and which point, it will fail back on the home host. Use the <b>LIST UTILITIES</b> command with the SHOW DETAIL option from any active member to monitor recovery progress, and to see if crash recovery is complete for all databases. The member does not accept any new connections, nor does it process any transactions. In-doubt transactions might still exist. |
|           | ERROR                | DB2 cluster services could not automatically restart the member, either on its home host or on any other host in the DB2 pureScale instance. When the member reflects an ERROR state, the ALERT field is always set to YES, indicating intervention and investigation is required from the administrator. DB2 cluster services no longer attempts to restart the member once it is in the ERROR state unless the alert has been cleared.  |

## Alerts for hosts, members and cluster caching facilities

In addition to returning state information, the commands that query the status of the components in a DB2 pureScale environment also return alert information. The possible values for alerts for all components is either YES or NO. Generally speaking, an alert value of NO is an indication that things are running normally. An alert value of YES indicates that there is a problem that might require manual intervention. In some cases, the alert conditions are temporary, and the alert field might clear itself, such as when a host is rebooted. In other cases, the alert field remains set until the administrator resolves the problem and manually resets the alert field using the **db2cluster** command with the **-clear -alert** options.

---

## Interpretation of status information

When you query hosts, members or cluster caching facilities for status information, the system presents state and alert information that tells you about the status of the various components in your DB2 pureScale environment. When problems arise, you generally need to examine both states and alerts to understand what is happening in the system.

The *state* of a host, member or cluster caching facility (also known as a CF) reflects its operational status. When everything is operating normally, the values reported for the state of hosts, members and cluster caching facilities (also known as CFs) can give you a general idea of the status of your system. For example, a status of **RESTARTING**, or **WAITING\_FOR\_FAILBACK** on a member does not itself indicate that there is a problem. There might be several valid reasons why a member is

failing over to a new host, or restarting on its home host, such as when hosts are taken offline for maintenance. If a member is failing over on a frequent, repeated basis, there might be a problem that warrants further investigation.

An *alert* for a host, member or CF is an indication that a problem exists that might require investigation or intervention. Looking at alerts in the context of the state of a given system component can reveal additional information about the source of the problem. The sections that follow outline the various combinations of state and alert information that you might encounter for hosts, members or cluster caching facilities, and how to interpret different combinations of states and alerts.

**Remember:** The completeness of state and alert information returned by the interfaces that report on this information depends on the following factors:

- The type of instance in which the table function, administrative view, or command is being run (for example, DB2 pureScale instances or other DB2 instances)
- Whether a supported cluster manager is employed in that instance. All DB2 pureScale Feature deployments use a cluster manager.

See “Differences in reporting for data-sharing and environments other than DB2 pureScale environments” on page 1443 for details.

## Host status

You can view information about the hosts in a DB2 pureScale environment using a number of different interfaces. One such interface is the `DB2_CLUSTER_HOST_STATE` administrative view. For example, consider this SQL query:

```
SELECT varchar(HOSTNAME,10) AS HOST,  
       varchar(STATE,8) AS STATE,  
       varchar(INSTANCE_STOPPED,7) AS STOPPED,  
       ALERT  
FROM SYSIBMADM.DB2_CLUSTER_HOST_STATE
```

The output of running the preceding SQL statement would look like this:

| HOST  | STATE  | STOPPED | ALERT |
|-------|--------|---------|-------|
| HOSTD | ACTIVE | NO      | NO    |
| HOSTB | ACTIVE | NO      | NO    |
| HOSTA | ACTIVE | YES     | NO    |
| HOSTC | ACTIVE | NO      | NO    |

4 record(s) selected.

(In the preceding example, the `STOPPED` column corresponds to the `INSTANCE_STOPPED` column returned by the administrative view.)

The values for the `state`, `instance_stopped` and `alert` columns can take on different values, depending on the conditions at any given time. The possible values are summarized in Table 2123 on page 1441.

Table 2123. Combinations of state, instance\_stopped, and alerts possible on a host system in a DB2 pureScale instance

| STATE    | INSTANCE_STOPPED | ALERT | Description   |
|----------|------------------|-------|---|
| ACTIVE   | NO               | NO    | The host is active and operating normally.  |
|          |                  | YES   | The host is active (that is, it responds to system commands), however there might be a problem preventing it from participating in the DB2 pureScale instance. For example, there might be a file system problem or a network communication issue, or the idle processes that the DB2 pureScale Feature requires for performing failovers might not be running. |
|          | YES              | NO    | The host is active. The instance has been stopped explicitly on this host by the administrator using the <b>db2stop instance on hostname</b> command  |
|          |                  | YES   | The host is active, however, an alert exists for the host that has not been cleared. The administrator has explicitly stopped the instance.   |
| INACTIVE | NO               | NO    | Not applicable. A host cannot be INACTIVE when both INSTANCE_STOPPED and ALERT are set to NO.   |
|          |                  | YES   | The host is not responding to system commands. The instance was not stopped explicitly by the administrator, however there is an alert. This combination of status information indicates the abnormal shutdown of a host. Such a shutdown might arise, for example, from a power failure on a host.   |
|          | YES              | NO    | Normal state when the instance has been stopped by the administrator. Such a combination of status information might arise when the host is being taken offline for the installation of software updates.   |
|          |                  | YES   | The host is not responding to system commands. An alert exists for the host that has not been cleared, but the instance was stopped explicitly by the administrator (that is, the system did not shut down abnormally).   |

**Tip:** You can see details about alerts using the DB2\_INSTANCE\_ALERTS administrative view.

## Member status

You can view member states and alerts using several different interfaces. One such interface is the DB2\_MEMBER administrative view. The DB2\_MEMBER administrative view shows status information for members in a DB2 pureScale instance. What follows is an example of how to use this administrative view to retrieve member status:

```
SELECT ID,
       varchar(STATE,21) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

The values for the state, and alert columns can take on different values, depending on the conditions at any given time. The possible values are summarized in Table 2124 on page 1442.

Table 2124. Combinations of state, and alerts possible for members in a DB2 pureScale instance

| STATE                | ALERT | Description   |
|----------------------|-------|---|
| STARTED              | NO    | The member is started in the instance and is operating normally.  |
|                      | YES   | The member is started in the instance. However, at some point, there was an unsuccessful attempt to fail over to another host. Since that unsuccessful attempt to fail over, the member was able to fail over successfully to another host, or it has failed back to its home host. If the member is running on its home host, it is running normally; if it is running on a guest host, it is running in light mode. Either way, investigate the alert to determine what happened. |
| STOPPED              | NO    | The member has been stopped by the administrator using the <b>db2stop</b> command.  |
|                      | YES   | The member has been stopped by the administrator using the <b>db2stop</b> command, however, the alert field has not yet been cleared.   |
| RESTARTING           | NO    | The member is starting.   |
|                      | YES   | The member is starting. However, at some point, there was an unsuccessful attempt to start the member on the home host or to fail over to another host. The alert field has not yet been cleared.   |
| WAITING_FOR_FAILBACK | NO    | The member is running in light mode on a guest host, and is waiting to fail back to the home host. You might want to examine the status of the home host to see if anything is preventing the member from failing back to the home host (for example, a failed network adapter).  |
|                      | YES   | An attempt to restart the member on the home host might have failed, automatic failback is disabled, or crash recovery might have failed. You need to resolve the problem and clear the alert manually before the member can automatically fail back to its home host. If automatic failback is disabled, manually clear the alert and enable automatic failback using the <b>db2cluster</b> command.   |
| ERROR                | YES   | DB2 cluster services was not able to start the member on any host. You need to resolve the problem and clear the alert manually before attempting to restart the instance.  |

**Tip:** You can see details about alerts using the DB2\_INSTANCE\_ALERTS administrative view.

### cluster caching facility status

The DB2\_GET\_INSTANCE\_INFO table function lets you retrieve status information for members in a DB2 pureScale instance. One of the benefits of the table function is that you can pass parameters to it to narrow the scope of the results returned. For example, to retrieve information about CFs in a DB2 pureScale instance, you can construct a query such as:

```
SELECT ID,
       varchar(STATE,17) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM TABLE(DB2_GET_INSTANCE_INFO(NULL, '', '', 'CF', NULL))
```

The values for the state, and alert columns can take on different values, depending on the conditions at any given time. The possible values are summarized in Table 2125 on page 1443.

Table 2125. Combinations of state, and alerts possible for cluster caching facilities in a DB2 pureScale instance

| STATE                | ALERT | Description   |
|----------------------|-------|---|
| STOPPED              | NO    | The cluster caching facility (also known as a CF) has been manually stopped using the <b>db2stop</b> command.   |
|                      | YES   | There has been an unsuccessful attempt by the CF to become the primary CF. The cluster caching facility has been manually stopped in the instance by the administrator using the <b>db2stop</b> command.  |
| RESTARTING           | NO    | The CF is restarting, either as a result of the <b>db2start</b> command, or after a primary CF failure.   |
|                      | YES   | The CF is restarting, however, there is a pending alert from a previous failed attempt by the CF to take on the primary role that must be cleared manually.   |
| BECOMING_PRIMARY     | NO    | The CF will take on the role of primary CF if there is no other primary CF already running in the instance.   |
|                      | YES   | Not applicable. The CF cannot attempt to take on the primary role with an alert condition set.  |
| PRIMARY              | NO    | The CF has taken on the role of primary CF and is operating normally.   |
|                      | YES   | Not applicable. The CF cannot be acting as the primary CF with an alert condition set.  |
| CATCHUP( <i>n</i> %) | NO    | This non-primary CF is in the process of copying information from the primary CF required for it to operate in PEER mode.<br><br><b>Note:</b> When you view the status of the non-primary CF using the command <b>db2instance -list</b> , it will be in <b>CATCHUP</b> state until a connection is made to the database. Once the first connection is made, the process of copying data from the primary CF begins. |
|                      | YES   | This non-primary CF is in the process of copying information from the primary CF required for it to operate in PEER mode. There is a pending alert from a previous failed attempt by this CF to take on the primary role that must be cleared manually.   |
| PEER                 | NO    | This non-primary CF is ready to assume the role of primary CF if the current primary CF fails.  |
|                      | YES   | This non-primary CF is ready to assume the role of primary CF if the current primary CF fails. There is a pending alert from a previous failed attempt by this CF to take on the primary role that must be cleared manually.  |
| ERROR                | YES   | The CF could not be started on any host in the instance. You need to resolve the problem and clear the alert manually before attempting to restart the instance.  |

**Tip:** You can see details about alerts using the DB2\_INSTANCE\_ALERTS administrative view.

### Differences in reporting for data-sharing and environments other than DB2 pureScale environments

All the various table functions, administrative views and commands that report status data for hosts, members and cluster caching facilities can be used outside of a DB2 pureScale instance. However, the results returned by these interfaces might be different from what you see in a DB2 pureScale instance.

In a configuration that uses a clustered file system with a supported cluster manager (CM) (a configuration sometimes known as "integrated High Availability" or "integrated HA") the results returned for most of these status-reporting interfaces will resemble what you see in a DB2 pureScale instance. One exception is when retrieving information about hosts in your instance using the DB2\_GET\_CLUSTER\_HOST\_STATE table function or the DB2\_CLUSTER\_HOST\_STATE administrative view. Outside of a DB2 pureScale instance with integrated HA, neither of these interfaces will return the INSTANCE\_STOPPED column. The results of a query that uses the DB2\_CLUSTER\_HOST\_STATE administrative view, for example, resemble those shown in Figure 20

| HOSTNAME | STATE  | INSTANCE_STOPPED | ALERT |
|----------|--------|------------------|-------|
| HOSTA    | ACTIVE | -                | NO    |
| HOSTB    | ACTIVE | -                | NO    |
| HOSTC    | ACTIVE | -                | NO    |
| HOSTD    | ACTIVE | -                | NO    |

Figure 20. Results returned by the DB2\_CLUSTER\_HOST\_STATE administrative view outside of a DB2 pureScale instance with a cluster manager.

Another exception is any interface that specifically reports on status for cluster caching facilities. Outside of a DB2 pureScale environment, there are no cluster caching facilities, so there is no status to report. For example, the DB2\_CF administrative view returns results similar to the following in an environment other than a DB2 pureScale environment:

| ID                    | CURRENT_HOST | STATE | ALERT |
|-----------------------|--------------|-------|-------|
| 0 record(s) selected. |              |       |       |

Figure 21. Results returned by the DB2\_CF administrative view outside of a DB2 pureScale instance.

When the status-reporting interfaces are used in an instance without a CM, no status or alert information is returned at all. For example, the results of a query that uses the DB2\_CLUSTER\_HOST\_STATE administrative view resemble those in Figure 22

| HOSTNAME | STATE | INSTANCE_STOPPED | ALERT |
|----------|-------|------------------|-------|
| HOSTA    | -     | -                | -     |
| HOSTB    | -     | -                | -     |
| HOSTC    | -     | -                | -     |
| HOSTD    | -     | -                | -     |

4 record(s) selected.

Figure 22. Results returned by the DB2\_CLUSTER\_HOST\_STATE administrative view outside of a DB2 pureScale instance without a cluster manager.

---

## Examples: Viewing the status of hosts, members and cluster caching facilities

The topics that follow include examples of using the various interfaces to view the status of the components of your DB2 pureScale instance.

## Viewing status information for hosts in a DB2 pureScale instance

You can retrieve basic information that shows the overall status of hosts in a DB2 pureScale instance. This information tells you whether the host is active or not, if the instance is running on the host, and whether there are any alerts that require investigation.

### About this task

Viewing the status of the hosts in a DB2 pureScale instance is one of the first places to start to get an overall view of the status of the instance.

One way to retrieve this status is using the `DB2_CLUSTER_HOST_STATE` administrative view. This view returns status information for all hosts in the instance. You can also use the following interfaces to retrieve information about host status:

- `DB2_GET_CLUSTER_HOST_STATE` table function. This approach is useful if you want to query the status of a particular host, as the table function accepts a host ID as a parameter.
- `LIST INSTANCE` command.
- `db2instance` command, with the `-list` parameter

### Procedure

To view the status of the hosts in a DB2 pureScale instance:

1. Formulate an SQL statement using either the `DB2_CLUSTER_HOST_STATE` administrative view, or the `DB2_GET_CLUSTER_HOST_STATE` table function. This example uses the administrative view:

```
SELECT varchar(HOSTNAME,10) AS HOST,  
       varchar(STATE,8) AS STATE,  
       varchar(INSTANCE_STOPPED,7) AS STOPPED,  
       ALERT  
FROM SYSIBMADM.DB2_CLUSTER_HOST_STATE
```

2. Run the query.

### Results

The output of running the preceding SQL statement would look like this:

| HOST  | STATE  | STOPPED | ALERT |
|-------|--------|---------|-------|
| HOSTD | ACTIVE | NO      | NO    |
| HOSTB | ACTIVE | NO      | NO    |
| HOSTA | ACTIVE | YES     | NO    |
| HOSTC | ACTIVE | NO      | NO    |

4 record(s) selected.

In this example, there are 4 hosts in this instance. Three are active, which just means that the systems are powered on, and are able to respond to operating system commands. One host is stopped, which means that the instance has been stopped on that host explicitly by the administrator. There are no alerts that require investigation.

### Example

*Retrieving host status using the `DB2_GET_CLUSTER_HOST_STATE` table function*

The DB2\_GET\_CLUSTER\_HOST\_STATE table function also lets you retrieve status information about hosts in a DB2 pureScale instance. One of the benefits of the table function is that you can pass parameters to it to narrow the scope of the results returned. For example, to retrieve information about the host HOSTD in the DB2 pureScale instance, construct a query like the following:

```
SELECT varchar(HOSTNAME,10) as HOST,
       varchar(STATE,10) AS STATE,
       ALERT
FROM TABLE(DB2_GET_CLUSTER_HOST_STATE('HOSTD'))
```

Results:

| HOST  | STATE  | ALERT |
|-------|--------|-------|
| HOSTD | ACTIVE | NO    |

1 record(s) selected.

## Viewing status information for members and cluster caching facilities in a DB2 pureScale instance

You can view details about the operational status of members and cluster caching facilities (also known as CFs) in a DB2 pureScale instance, such as the role played by CFs (for example, primary or peer), and whether or members have failed over to a different host.

### About this task

The example presented in this task shows how to retrieve information about the status of members and cluster caching facilities in a DB2 pureScale instance using the **db2instance** system command. The benefit of using a system command is that a database connection is not required. However, the command must be run from a host that is a member (not a CF) in the instance.

### Procedure

To retrieve status information about the members and CFs in a DB2 pureScale instance using the **db2instance** command, enter the command at the system prompt of one of the members in the instance, with the **-list** option:

```
db2instance -list
```

The **db2instance** command returns information like the following query (output has been slightly compressed for presentation purposes):

| ID  | TYPE   | STATE   | HOME_HOST | CURRENT_HOST | ALERT | PARTITION_NUMBER | LOGICAL_PORT | NETNAME   |
|-----|--------|---------|-----------|--------------|-------|------------------|--------------|-----------|
| 0   | MEMBER | STARTED | HOSTA     | HOSTA        | NO    | 0                | 0            | OSTA-ib0  |
| 1   | MEMBER | STARTED | HOSTB     | HOSTB        | NO    | 0                | 0            | HOSTB-ib0 |
| 2   | MEMBER | STARTED | HOSTC     | HOSTC        | NO    | 0                | 0            | HOSTC-ib0 |
| 128 | CF     | PRIMARY | HOSTD     | HOSTD        | NO    | -                | 0            | HOSTD-ib0 |
| 129 | CF     | PEER    | HOSTE     | HOSTE        | NO    | -                | 0            | HOSTE-ib0 |

| HOSTNAME | STATE  | INSTANCE_STOPPED | ALERT |
|----------|--------|------------------|-------|
| HOSTA    | ACTIVE | NO               | NO    |
| HOSTC    | ACTIVE | NO               | NO    |
| HOSTD    | ACTIVE | NO               | NO    |
| HOSTE    | ACTIVE | NO               | NO    |
| HOSTB    | ACTIVE | NO               | NO    |

## Results

The results returned depend on the structure of your DB2 pureScale instance. In this example, the report shows that:

- Hosts HOSTA through HOSTC are configured as members
- Each member is started, and is running on its own home host
- There are 2 cluster caching facilities CFs running on hosts HOSTD and HOSTE
- The primary CF is running on HOSTD; another CF is running on HOSTE in peer mode, indicating that it is ready to take over the responsibilities of the primary CF in the event of a primary CF failure.

## Example

You can also retrieve status information for members and cluster caching facilities using the following interfaces:

- DB2\_MEMBER or DB2\_CF administrative view
- DB2\_GET\_INSTANCE\_INFO table function
- **LIST INSTANCE** command-line processor (CLP) command
- **db2cluster** system command.

The examples that follow illustrate the use of some of these interfaces.

*Example 1: Retrieving status information using the DB2\_MEMBER administrative view*

The DB2\_MEMBER administrative view shows status information for members in a DB2 pureScale instance. What follows is an example of how to use this administrative view to retrieve member status:

```
SELECT ID,  
       varchar(STATE,21) AS STATE,  
       varchar(HOME_HOST,10) AS HOME_HOST,  
       varchar(CURRENT_HOST,10) AS CUR_HOST,  
       ALERT  
FROM SYSIBMADM.DB2_MEMBER
```

Results:

| ID | STATE                       | HOME_HOST | CUR_HOST     | ALERT |
|----|-----------------------------|-----------|--------------|-------|
| 0  | <b>WAITING_FOR_FAILBACK</b> | HOSTA     | <b>HOSTB</b> | NO    |
| 1  | STARTED                     | HOSTB     | HOSTB        | NO    |
| 2  | STARTED                     | HOSTC     | HOSTC        | NO    |

3 record(s) selected.

In this example, member 0 has failed on its home host and has failed over to HOSTB. Member 0 is waiting to fail back to its home host, HOSTA.

*Example 2: Retrieving status information for CFs using the DB2\_GET\_INSTANCE\_INFO table function*

The DB2\_GET\_INSTANCE\_INFO table function lets you retrieve status information for members in a DB2 pureScale instance. One of the benefits of the table function is that you can pass parameters to it to narrow the scope of the results returned. For example, to retrieve information about CFs in a DB2 pureScale instance, you can construct a query such as:

```
SELECT ID,  
       varchar(STATE,17) AS STATE,  
       varchar(HOME_HOST,10) AS HOME_HOST,
```

```

        varchar(CURRENT_HOST,10) AS CUR_HOST,
        ALERT
FROM TABLE(DB2_GET_INSTANCE_INFO(NULL, '', '', 'CF', NULL))

```

Results:

| ID  | STATE            | HOME_HOST | CUR_HOST | ALERT |
|-----|------------------|-----------|----------|-------|
| 128 | RESTARTING       | HOSTD     | HOSTD    | NO    |
| 129 | BECOMING_PRIMARY | HOSTE     | HOSTE    | NO    |

1 record(s) selected.

In this example, the CF with the host ID of 128 has failed. The CF with the host ID of 129 is in the process of taking over as the primary CF.

*Example 3: Investigating alerts reported with the **db2instance -list** command.*

In this example, the results of running the **db2instance -list** command are as follows:

```

$ db2instance -list
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT      PARTITION_NUMBER      LOGICAL_PORT      NETNAME
-----
0       MEMBER    STARTED    HostA           HostA             NO         0                     0                 -
1       MEMBER    STARTED    HostB           HostB             NO         0                     1                 -
2       MEMBER    STARTED    HostC           HostC             NO         0                     2                 -
128    CF        ERROR      HostD           HostD             YES        -                     0                 -
129    CF        ERROR      HostE           HostE             YES        -                     0                 -

HOSTNAME      STATE      INSTANCE_STOPPED      ALERT
-----
HostA          ACTIVE    NO                     NO
HostB          ACTIVE    NO                     NO
HostC          ACTIVE    NO                     NO
HostD          ACTIVE    NO                     YES
HostE          ACTIVE    NO                     YES

```

There is currently an alert for a member, CF, or host in the data-sharing instance. For more information on the alert, its impact, and how to clear it, run the following command: 'db2cluster -cm -list -alert'

In this example, there are alerts for both cluster caching facilities in the instance. Also, the state of the CFs appear as ERROR. As the message at the end of the report suggests, you can use the **db2cluster** command with the **-cm -list -alert** options to view more information about the alerts:

```
$db2cluster -cm -list -alert
```

Alert: CF '128' failed to start the PRIMARY role on host 'HostD'. Check the cadiag\*.log for failures related to CF '128' for more information.

Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.

Impact: CF '128' on host 'HostD' will be unavailable to service requests from DB2 members until the alert is cleared.

*Example 4: Member alert, a member failed to start in the DB2 pureScale instance*

In this example, the results of running the **db2instance -list** command are as follows:

```

$ db2instance -list
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT      PARTITION_NUMBER      LOGICAL_PORT      NETNAME
-----
0       MEMBER    ERROR      HostA           HostA             YES        0                     0                 -
1       MEMBER    STARTED    HostB           HostB             NO         0                     1                 -
2       MEMBER    STARTED    HostC           HostC             NO         0                     2                 -
128    CF        PRIMARY    HostD           HostD             NO         -                     0                 -
129    CF        PEER      HostE           HostE             NO         -                     0                 -

HOSTNAME      STATE      INSTANCE_STOPPED      ALERT
-----
HostA          ACTIVE    NO                     NO
HostB          ACTIVE    NO                     NO
HostC          ACTIVE    NO                     NO
HostD          ACTIVE    NO                     NO
HostE          ACTIVE    NO                     NO

```

There is currently an alert for a member, CF, or host in the data-sharing instance. For more information on the alert, its impact, and how to clear it, run the following command: 'db2cluster -cm -list -alert'

In this example, a member failed to start in the DB2 pureScale instance. Running the **db2cluster** command with the **-cm -list -alert** options recommends an action to take and outlines the impact of this failure in the

## DB2 pureScale instance.

```
$db2cluster -cm -list -alert
```

Alert: DB2 member '0' failed to start on its home host 'HostA'. The cluster manager will attempt to restart the DB2 member in restart light mode on another host. Check the db2diag.log for messages concerning failures on host 'HostA' for member '0'."

Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.

Impact: DB2 member '%0' will not be able to service requests until this alert has been cleared and the DB2 member returns to its home host.

### Example 5: CF error, the secondary CF failed CATCHUP phase

In this example, the results of running the **db2instance -list** command are as follows:

```
$ db2instance -list
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
-----
0       MEMBER    STARTED    HostA           HostA              NO     0                  0              -
1       MEMBER    STARTED    HostB           HostB              NO     0                  1              -
2       MEMBER    STARTED    HostC           HostC              NO     0                  2              -
128    CF        PRIMARY    HostD           HostD              NO     -                  0              -
129    CF        ERROR      HostE           HostE              YES    -                  0              -

HOSTNAME      STATE      INSTANCE_STOPPED  ALERT
-----
HostA         ACTIVE    NO                NO
HostB         ACTIVE    NO                NO
HostC         ACTIVE    NO                NO
HostD         ACTIVE    NO                NO
HostE         ACTIVE    NO                NO
```

There is currently an alert for a member, CF, or host in the data-sharing instance. For more information on the alert, its impact, and how to clear it, run the following command: 'db2cluster -cm -list -alert'

In this example, the secondary CF failed CATCHUP phase. Running the **db2cluster** command with the **-cm -list -alert** options recommends an action to take and outlines the impact of this failure in the DB2 pureScale instance.

```
$db2cluster -cm -list -alert
```

Alert: CF '129' failed to complete CATCHUP on host 'HostE'. Check the db2diag.log for failure messages pertaining to CATCHUP on CF '129'.

Action: Contact IBM support to determine the reason for the failure. To re-attempt CATCHUP, restart the failed CF with the commands: 'db2stop 129; db2start 129'. This alert will clear itself when the CF is restarted.

Impact: CF '129' on host 'HostE' will not be available until it can undergo CATCHUP successfully.

### Example 6: Host alert, the host "HostA" lost a network connection.

In this example, the results of running the **db2instance -list** command are as follows:

```
$ db2instance -list
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
-----
0       MEMBER    WAITING_FOR_FAILBACK  HostA           HostA              NO     0                  0              -
1       MEMBER    STARTED    HostB           HostB              NO     0                  1              -
2       MEMBER    STARTED    HostC           HostC              NO     0                  2              -
128    CF        PRIMARY    HostD           HostD              NO     -                  0              -
129    CF        PEER       HostE           HostE              NO     -                  0              -

HOSTNAME      STATE      INSTANCE_STOPPED  ALERT
-----
HostA         INACTIVE    NO                YES
HostB         ACTIVE    NO                NO
HostC         ACTIVE    NO                NO
HostD         ACTIVE    NO                NO
HostE         ACTIVE    NO                NO
```

There is currently an alert for a member, CF, or host in the data-sharing instance. For more information on the alert, its impact, and how to clear it, run the following command: 'db2cluster -cm -list -alert'.

In this example, the host "HostA" lost a network connection. Running the **db2cluster** command with the **-cm -list -alert** options recommends an action to take and outlines the impact of this failure in the DB2 pureScale instance.

```
$db2cluster -cm -list -alert
```

Alert: Host 'HostA' is INACTIVE. Ensure the host is powered on and connected to the network.

Action: This alert will clear itself when the host is ACTIVE.

Impact: While the host is INACTIVE, the DB2 members on this host will be in restart light mode on other hosts and will be in the WAITING\_FOR\_FAILBACK state. Any CF defined on the host will not be able to start, and the host will not be available as a target for restart light.

## Checking restart status for members

If you know a member has failed, perhaps because of a power loss or other hardware problem that you have since corrected, you might want to know whether it has restarted successfully.

### About this task

You can use the DB2\_MEMBER administrative view to examine the operational status of all members in a DB2 pureScale instance. You can also use the DB2\_GET\_INSTANCE\_INFO table function, which provides options for querying specific hosts.

The process for checking member restart status is exactly as is shown in Example 1 of “Viewing status information for members and cluster caching facilities in a DB2 pureScale instance” on page 1446. Specifically, formulate an SQL query that uses the DB2\_MEMBER administrative view (or the DB2\_GET\_INSTANCE\_INFO table function) to retrieve values for the following columns:

- ID
- HOME\_HOST
- CURRENT\_HOST
- STATE
- ALERT

### Procedure

1. Formulate the SQL query using whichever interface you prefer. This example uses the DB2\_MEMBER administrative view:

```
SELECT ID,  
       varchar(STATE,21) AS STATE,  
       varchar(HOME_HOST,10) AS HOME_HOST,  
       varchar(CURRENT_HOST,10) AS CUR_HOST,  
       ALERT  
FROM SYSIBMADM.DB2_MEMBER
```

2. Run the query. The results returned will look like the following:

| ID | STATE   | HOME_HOST | CUR_HOST | ALERT |
|----|---------|-----------|----------|-------|
| 0  | STARTED | HOSTA     | HOSTA    | NO    |
| 1  | STARTED | HOSTB     | HOSTB    | NO    |
| 2  | STARTED | HOSTC     | HOSTC    | NO    |

3 record(s) selected.

In the previous example, all members are running on their own hosts with no alerts.

### Results

When looking at the restart status for the members, check that:

- The value for the STATE column is either RESTARTING or STARTED. The former is an indication that the member is in the process of being restarted; the

latter indicates that it has successfully restarted. If the state is `RESTARTING`, check the status again in a few minutes to see if the state has changed to `STARTED`.

- The value for `CUR_HOST` is the same as the value for `HOME_HOST`. This indicates that the member is running on its home host.
- There is no `YES` value in the alert column for the member you are interested in.

If `CUR_HOST` is different than `HOME_HOST`, or if the state has not moved beyond `RESTARTING`, or remains as `WAITING_FOR_FAILBACK`, or if there is a `YES` value in the alert column, then there might be a problem that requires further investigation.

## Example

### *Example 1: Failed member in the process of restarting*

In this example, member 0 is in the process of restarting on its home host, `HOSTA`.

| ID | STATE             | HOME_HOST | CUR_HOST | ALERT |
|----|-------------------|-----------|----------|-------|
| 0  | <b>RESTARTING</b> | HOSTA     | HOSTA    | NO    |
| 1  | STARTED           | HOSTB     | HOSTB    | NO    |
| 2  | STARTED           | HOSTC     | HOSTC    | NO    |

3 record(s) selected.

To see if the restart is ultimately successful, run the query again in a few seconds.

### *Example 2: Failed member that is not able to restart*

In this example, member 0 is waiting to fail back to its home host. Currently, it is running in light mode on `HOSTB`.

| ID | STATE                       | HOME_HOST | CUR_HOST | ALERT |
|----|-----------------------------|-----------|----------|-------|
| 0  | <b>WAITING_FOR_FAILBACK</b> | HOSTA     | HOSTB    | NO    |
| 1  | STARTED                     | HOSTB     | HOSTB    | NO    |
| 2  | STARTED                     | HOSTC     | HOSTC    | NO    |

3 record(s) selected.

In this case, you might want to check the host status for `HOSTA` to see if there is an issue. Using the `DB2_CLUSTER_HOST_STATE` administrative view might return the following results:

| HOST  | STATE    | STOPPED | ALERT      |
|-------|----------|---------|------------|
| HOSTD | ACTIVE   | NO      | NO         |
| HOSTB | ACTIVE   | NO      | NO         |
| HOSTA | INACTIVE | NO      | <b>YES</b> |
| HOSTC | ACTIVE   | NO      | NO         |

4 record(s) selected.

This report shows that there is an alert on `HOSTA`, and that the host is inactive. However, the instance was not stopped using the `db2stop` command. Further investigation might an incident such as a loss of power to this host. Once the problem with the host is resolved, check the restart status again to see if the member is able to restart.

## Viewing details for an alert

If one of the members or cluster caching facilities reports an alert, you can view more information about the alert using the `DB2_INSTANCE_ALERTS` administrative view. Alternatively, you can use the `db2cluster` command with the `-cm -list -alert` parameters.

### About this task

This task assumes that you already determined an alert was raised on one of the hosts in your DB2 pureScale instance. For example, the `LIST INSTANCE` command might have shown an alert for one of your members:

| ID  | TYPE   | STATE   | HOME_HOST | CURRENT_HOST | ALERT |
|-----|--------|---------|-----------|--------------|-------|
| 0   | MEMBER | STARTED | hostA     | hostA        | YES   |
| 1   | MEMBER | STARTED | hostB     | hostB        | NO    |
| 2   | MEMBER | STARTED | hostC     | hostC        | NO    |
| 3   | MEMBER | STARTED | hostD     | hostD        | NO    |
| 128 | CF     | PRIMARY | hostE     | -            | NO    |
| 129 | CF     | PEER    | hostF     | -            | NO    |

| HOSTNAME | STATE  | INSTANCE_STOPPED | ALERT |
|----------|--------|------------------|-------|
| hostA    | ACTIVE | NO               | NO    |
| hostB    | ACTIVE | NO               | NO    |
| hostC    | ACTIVE | NO               | NO    |
| hostD    | ACTIVE | NO               | NO    |
| hostE    | ACTIVE | NO               | NO    |
| hostF    | ACTIVE | NO               | NO    |

In this case, member 0 is showing an alert.

### Procedure

To find more information about the alerts in your instance:

1. Formulate an SQL statement that uses the `DB2_INSTANCE_ALERTS` administrative view:

```
SELECT * FROM SYSIBMADM.DB2_INSTANCE_ALERTS
```

2. Run the SQL statement.

### Results

Depending on the nature of the problem on member 0, the information that the `DB2_INSTANCE_ALERTS` administrative view returns varies. You might, for example, receive a message like the one that follows:

```
MESSAGE
-----
Could not restart light DB2 member '0' on hosts 'hostA'. Check the db2diag.log for messages concerning a
restart light or database crash recovery failure on the indicated hosts for DB2 member '0'.

ALERT_ACTION
-----
This alert must be cleared manually with the command: 'db2cluster -clear -alert -member 0'

IMPACT
-----
DB2 member '0' will not be able to restart light on host 'hostC' until this alert has been cleared.
```

### Example

*Example 1: Using the `db2cluster` command to view alert information*

In this example, the `db2instance -list` command returns the following information:

```

$ db2instance -list
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
-----
0      MEMBER    ERROR      HostA          HostA             YES    0                 0             -
1      MEMBER    STARTED    HostB          HostB             NO     0                 1             -
2      MEMBER    STARTED    HostC          HostC             NO     0                 2             -
128    CF        PRIMARY    HostD          HostD             NO     -                 0             -
129    CF        PEER       HostE          HostE             NO     -                 0             -

HOSTNAME      STATE      INSTANCE_STOPPED  ALERT
-----
HostA         ACTIVE     NO                YES
HostB         ACTIVE     NO                NO
HostC         ACTIVE     NO                NO
HostD         ACTIVE     NO                NO
HostE         ACTIVE     NO                NO

```

There is currently an alert for a member, CF, or host in the data-sharing instance. For more information on the alert, its impact, and how to clear it, run the following command: 'db2cluster -cm -list -alert'

From the information contained in Table 2124 on page 1442, you can see that the output of **db2instance -list** is showing that member 0 was not able to start on any host. (If it had started on guest host, then the status would be STARTED, and the current host would show the name of the host it was running on.)

Using the **db2cluster -cm -list -alert** command shows the following message:

```
$ db2cluster -cm -list -alert
```

```
Alert: DB2 member '0' failed to start on its home host 'Host A'. The cluster manager will
attempt to restart the DB2 member
in restart light mode on another host. Check the db2diag.log for messages concerning failures
on hosts 'HostA' for member '0'.
```

```
Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.
```

```
Impact: DB2 member '0' will not be able to service requests until this alert has been cleared
and the DB2 member returns to its home host.
```

## What to do next

Perform the investigation or action specified in the information returned by the DB2\_INSTANCE\_ALERTS administrative view or the **db2cluster** command.



---

## Chapter 13. Event and real-time database and system monitoring in a DB2 pureScale environment

In addition to viewing the overall status of the components of a DB2 pureScale instance, you can examine specific aspects of the operation of cluster caching facilities and members using the DB2 monitoring infrastructure. You can use monitoring table functions and administrative views to display this information. You can also use selected event monitors to capture events as they occur.

DB2 V9.7 introduced a number of enhancements to the monitoring infrastructure for the DB2 product. One of these enhancements was a set of table functions that provide access to hundreds of *in-memory* monitor elements that you can use query the state of your database environment at a specific point in time. Other enhancements included improved event monitors for capturing information about such things as locking, units of work, and activities as they occur.

The DB2 pureScale Feature extends the monitoring capabilities built into the DB2 database with monitor elements that you can use to view data that describes specific aspects of the operation of cluster caching facilities (also known as CFs) and members in a DB2 pureScale instance. However, there are some differences between monitoring in DB2 pureScale instances and other DB2 instances to be aware of, including:

- “The ability to monitor CFs in addition to DB2 members”
- “How monitor elements in a DB2 pureScale instance are reported” on page 1456
- The effects of component failure on monitor element reporting.

### The ability to monitor CFs in addition to DB2 members

CFs, with the different role they play as compared to members in a DB2 pureScale environment introduce additional monitoring needs. For example, in DB2 instances other than DB2 pureScale instances, you might be interested in monitoring for buffer pool hit ratios, which represents the number of pages that are found in memory, as compared to the number of pages that must be read from disk. Higher buffer pool hit ratios are, generally speaking, a reflection of better performance. The higher performance is because there is less I/O involved in bringing needed pages into memory. In a DB2 pureScale environment, all physical page reads from disk are performed by the members, but only after they first check with the CF to see if the group buffer pool has a record of any other member with a valid page that they can use. Thus, whereas you might be accustomed to tuning only local buffer pools in a DB2 environment other than a DB2 pureScale environment, monitoring buffer pool hit ratios in the group buffer pool of the CF is also important in a DB2 pureScale environment. The more times pages can be found in either a local or group buffer pool (GBP), the fewer times they must be read in from disk.

In addition to the GBP, the global lock manager (GLM) is another component of the CF that you can monitor. The GLM manages locking of objects across all the members in a DB2 pureScale instance. The DB2 pureScale Feature adds monitor elements that you can use to monitor locking between members.

## How monitor elements in a DB2 pureScale instance are reported

In general, the mechanics of monitoring in a DB2 pureScale instance are similar to the mechanics of monitoring in other DB2 instances. For example, the `MON_GET_TABLESPACE` table function, which returns information about table spaces in a database, works similarly in both DB2 pureScale instance and other DB2 instances. In a DB2 pureScale instance, the scope of some monitor elements is limited to a specific member, while the scope of others is global, across all members. For example, the data from monitor elements such as **direct\_reads**, or **pool\_data\_l\_reads** are specific to read activity performed by a member. By comparison, monitor elements such as **tbasp\_total\_pages**, which represent physical attributes of a table space is the same across all members, because all members share the same table space. For example, consider the following query:

```
SELECT VARCHAR(TBSP_NAME, 30) AS TBSP_NAME,  
       MEMBER, POOL_DATA_L_READS,  
       TBSP_TOTAL_PAGES  
FROM TABLE(MON_GET_TABLESPACE('USERSPACE1',-2))
```

The results of this query look like the following example:

| TBSP_NAME  | MEMBER | POOL_DATA_L_READS | TBSP_TOTAL_PAGES |
|------------|--------|-------------------|------------------|
| USERSPACE1 | 1      | 0                 | 4096             |
| USERSPACE1 | 2      | 0                 | 4096             |
| USERSPACE1 | 3      | 0                 | 4096             |
| USERSPACE1 | 0      | 36                | 4096             |

4 record(s) selected.

In this example, the number of logical reads from the local buffer pool for each member is different because each member performs its reads independently of other members; however the total pages for the table space is the same across all members, because all members are working from the same instance of `USERSPACE1`.

## Effects of component failure on monitor element reporting

If a host, member or CF in a DB2 pureScale environment fails, unless the entire DB2 pureScale instance is taken down, you can still retrieve monitor elements from the instance. However, the components that fail do not generate statistics. This fact is apparent if you are running a query such as the first example shown in “How monitor elements in a DB2 pureScale instance are reported,” where data from each member is shown individually. If you use a query that aggregates information across members, though, you might not notice that data from a member is missing.

Another thing to keep in mind is that if a member fails while monitor element data collection is taking place, the data collection process pauses until the communications problem with the failed member has been detected, or the TCP/IP timeout period has passed. In this situation, the data is still reported, however, there is no information from the failed member.

Finally, keep in mind that if a member fails, all the statistics accumulated in the monitor elements are reset to 0.

---

## Cluster caching facility memory and CPU usage monitoring overview

A basic indicator of the operational effectiveness of a cluster caching facility is the extent to which memory and the CPU are consistently used to their maximum configured capacity.

### Memory usage

Cluster caching facilities (also known as CFs) use different memory heaps for the following purposes:

#### Group buffer pool memory

Group buffer pool memory is used for the group buffer pool for the DB2 pureScale instance. If this type of memory is consistently used to the maximum configured capacity, it might have a negative effect on performance. However, the fact that memory might be used to capacity is not itself an indicator that performance might be affected. Check the hit rates for the group buffer pool to confirm that performance has degraded. Low hit rates coupled with high group buffer pool memory usage might be an indication that this type of memory needs to be increased. This type of memory is configured by the **cf\_gbp\_sz** configuration parameter.

#### Lock memory

Lock memory is used for managing page locks across the DB2 pureScale instance. If there is insufficient memory available for locks on the CF, one or both of the following conditions might arise:

- Lock escalation might take place, which reduces concurrency for the objects involved
- Requests for locks might be denied, resulting in the SQL0912 message being returned.

This type of memory is configured by the **cf\_lock\_sz** configuration parameter.

#### Shared Communication Area (SCA) memory

SCA memory contains database-wide information for tables, indexes, table spaces, and catalogs. Each database has its own SCA memory in the CF. It is allocated during the first database activation on any DB2 member, and is not freed until the database is dropped, or the CF is stopped. If table partitioning is used then the information required to synchronize the table partitioning data between the CF and the members is also stored in SCA memory.

If this type of memory is used to capacity, tables may fail to load, and an error is returned. This type of memory is configured by the **cf\_sca\_sz** configuration parameter.

#### Overall CF memory

Overall CF memory is the total amount of physical memory available to the CF. It is set by the **cf\_mem\_size** configuration parameter. The memory for the group buffer pool, locks and shared communication area are all allocated out of this pool of memory. For this reason, the total amount of the memory allocated for these specific types of memory must not exceed the amount of memory configured using the **cf\_mem\_size** configuration parameter.

By default, configuration of each of these types of memory is performed automatically. The DB2 pureScale Feature provides monitor elements that you can use to examine the amount of each of these types of memory that is currently in

use by the system. There are also related elements that you can use to determine what the maximum size is for each type of memory, and whether a memory resize operation is in progress.

In addition to monitor elements that report on the usage of specific types of CF memory, you can use the ENV\_CF\_SYS\_RESOURCES administrative view to examine the total amount of physical and virtual storage available to the CF.

## CPU load

CPU load on the CF is an indication of how heavily taxed its processors are. If you find that the processors on the host where the CF is running are working at maximum capacity most of the time, it might be an indication that the host the CF is running on is not powerful enough. You might want to add processors, or upgrade to a more powerful system.

You can view the overall CPU load for the host serving as the CF in a DB2 pureScale instance using the ENV\_CF\_SYS\_RESOURCES administrative view.

**Note:** The value reported for CPU load reflects the total usage of the CPU for actual processing performed by the CF, and for host processes other than processes from the CF.

## Monitor elements for viewing cluster caching facility memory usage

The IBM DB2 pureScale Feature provides a number of monitor elements that report on cluster caching facility memory usage.

### Monitor elements

The following monitor elements provide information about how various CF memory heaps are allocated and used:

- “configured\_cf\_gbp\_size - Configured cluster caching facility group buffer pool size monitor element” on page 723
- “current\_cf\_gbp\_size - Current cluster caching facility group buffer pool size monitor element” on page 764
- “target\_cf\_gbp\_size - Target cluster caching facility group buffer pool size monitor element” on page 1294
- “configured\_cf\_lock\_size - Configured cluster caching facility lock size monitor element” on page 723
- “current\_cf\_lock\_size - Current cluster caching facility lock size monitor element” on page 764
- “target\_cf\_lock\_size - Target cluster caching facility lock size monitor element” on page 1295
- “configured\_cf\_mem\_size - Configured cluster caching facility memory size monitor element” on page 724
- “current\_cf\_mem\_size - Current cluster caching facility memory size monitor element” on page 764
- “configured\_cf\_sca\_size - Configured cluster caching facility shared communications area size monitor element” on page 723
- “current\_cf\_sca\_size - Current cluster caching facility shared communications area size monitor element” on page 764

- “target\_cf\_sca\_size - Target cluster caching facility shared communications area size monitor element” on page 1295

**Tip:** In all cases, the values reported for each of these monitor elements is expressed in terms of 4k pages. So, for example, if you queried the **current\_gbp\_size** monitor element, and it returned a value of 350, then the actual amount of memory currently used for the GBP would be  $350 \times 4096$  bytes = 1,433,600 bytes.

For most of these types of memory, there are three monitor elements that you can query that represent different aspects of how the memory is configured.

#### **Current**

The current memory size (for example, `current_cf_gbp_size`, or `current_cf_mem_size`) represents the amount of that type of memory currently in use by the system.

#### **Configured**

The configured memory size (for example, `configured_cf_sca_size`, `configured_cf_mem_size`) represents the total amount of that type of memory that is currently configured by the database as the maximum. The value for current memory can never exceed that of configured memory.

#### **Target**

The target memory size (for example, `target_cf_sca_size`) represents a new configured maximum value for that type of memory. Usually, the target size is the same as the configured size. However, if the target and configured sizes differ, that means that that particular type of memory is undergoing an online change in its configured size. The process of allocating memory takes place over time. At any point during this resizing process, the configured memory represents the maximum amount of that type of memory that can be used at that specific point in time. Eventually, the configured memory becomes the same as target memory.

Refer to the reference topics for each monitor element to see what monitoring interfaces you can use to examine the data associated with that monitor element.

## **Retrieving information from cluster caching facility memory usage monitor elements**

You can use the `MON_GET_CF` table function to retrieve the various monitor elements that report on memory usage on the cluster caching facility in a DB2 pureScale instance.

### **About this task**

Understanding the extent to which memory in a cluster caching facility (also known as a CF) is used can help you to determine whether to adjust memory allocations. For example, if your cluster caching facility is using close to the maximum amount of allocated buffer pool memory, the hit rates for the group buffer pool might be lower than they could be. Or, if your lock memory is used to capacity, you might notice a higher-than-expected number of lock escalations.

### **Procedure**

To retrieve information about memory usage in a cluster caching facility:

1. Determine which monitor elements you want to retrieve. For example, if you want to view lock memory usage, you can choose from one or more of the following monitor elements:
  - current\_cf\_lock\_size
  - configured\_cf\_lock\_size
  - target\_cf\_lock\_size
2. Formulate a query using the MON\_GET\_CF table function. Using the example from step 1, the statement would look like the following example:

```
SELECT SUBSTR(HOST_NAME,1,10) AS HOST,
       ID as HOSTID,
       CURRENT_CF_LOCK_SIZE,
       CONFIGURED_CF_LOCK_SIZE,
       TARGET_CF_LOCK_SIZE
FROM TABLE( MON_GET_CF( NULL ) )
```

3. Run the query. Continuing with this example, the output from the preceding query would resemble the following example:

```
HOST      HOSTID CURRENT_CF_LOCK_SIZE CONFIGURED_CF_LOCK_SIZE
-----
HOSTA     128      133852                564224
HOSTB     129      133852                564224

TARGET_CF_LOCK_SIZE
-----
                564224
                564224
```

2 record(s) selected.

**Note:** The lock memory on both cluster caching facilities is typically the same, as one of the cluster caching facilities is duplexing its information and configuration to the other. You can check to see which of the two cluster caching facilities is the primary one by using one of the interfaces described in “Viewing status information for members and cluster caching facilities in a DB2 pureScale instance” on page 1446. Examples of two such interfaces are the **LIST INSTANCE** and **db2cluster** commands.

## Results

The preceding example shows that the current amount of lock memory used is 170,258 4k blocks, or 697,376,768 bytes. The maximum amount of lock memory available is 564,224 4k blocks, or 2,311,061,504 bytes.

## Example

*Example 1: Retrieving group buffer pool memory usage data.*

The following query displays information about the group buffer pool memory size for all cluster caching facilities on the system:

```
SELECT SUBSTR(HOST_NAME,1,20) AS HOST,
       ID as HOSTID,
       CURRENT_CF_GBP_SIZE,
       CONFIGURED_CF_GBP_SIZE,
       TARGET_CF_GBP_SIZE
FROM TABLE( MON_GET_CF(NULL) )
```

The following output is an example of what the preceding query returns:

| HOST  | HOSTID | CURRENT_CF_GBP_SIZE | CONFIGURED_CF_GBP_SIZE | TARGET_CF_GBP_SIZE |
|-------|--------|---------------------|------------------------|--------------------|
| HOSTA | 128    | 367611              | 500224                 | 500224             |
| HOSTB | 129    | 367611              | 500224                 | 500224             |

2 record(s) selected.

In this example, the current GBP size is 367,611 4k pages, or 1,505,734,656 bytes. The memory allocated to the GBP is 500,224 4k pages, or 2,048,917,504 bytes.

*Example 2: Retrieving shared communications area (SCA) memory usage data for a specific host.*

The following query displays information about the SCA memory size for the cluster caching facility with the ID of 128:

```
SELECT SUBSTR(HOST_NAME,1,8)AS HOST,
       ID as HOSTID,
       CURRENT_CF_SCA_SIZE,
       CONFIGURED_CF_SCA_SIZE,
       TARGET_CF_SCA_SIZE
FROM TABLE(MON_GET_CF(128))
```

The following output is an example of what the preceding query returns:

| HOST  | HOSTID | CURRENT_CF_SCA_SIZE | CONFIGURED_CF_SCA_SIZE | TARGET_CF_SCA_SIZE |
|-------|--------|---------------------|------------------------|--------------------|
| HOSTA | 128    | 43                  | 16128                  | 23280              |

1 record(s) selected.

In this example, the SCA memory currently being used is 43 4k pages. At the point these monitor elements were retrieved, the maximum size of the SCA memory was 16,128 4k pages. However, the configured and the target sizes are the different, which means that the SCA memory is undergoing a size increase from the previous maximum configured size to 23,280 pages.

## What to do next

In all the preceding examples, keep in mind that the values returned for memory usage provide only an overall view of memory usage. By themselves, they do not necessarily convey sufficient information to inform a decision about changing memory configuration. For example, the size of the group buffer pool (GBP), by itself, does not tell you whether it is large enough for the DB2 pureScale instance. In this case, consider using the monitor elements that report on buffer pool activity to calculate buffer pool hit rates. The hit rates can tell you whether you must adjust the size of your buffer pool. When it comes to lock memory, examining the number of lock escalations that take place gives you insight into whether enough lock memory has been allocated. A high rate of lock escalations can be an indicator that lock memory might need to be increased.

## Viewing cluster caching facility processor load

You can view the overall CPU load on the primary cluster caching facilities in a DB2 pureScale instance using the ENV\_CF\_SYS\_RESOURCES administrative view.

## Before you begin

You must be connected to a database running in a DB2 pureScale instance.

## About this task

The ENV\_CF\_SYS\_RESOURCES administrative view returns information for all cluster caching facilities (also known as CFs) in a DB2 pureScale instance. In instances where you have more than one CF configured, one to act as the primary and the others to serve as backup CFs running in PEER mode, the ENV\_CF\_SYS\_RESOURCES administrative view returns information for all CFs.

**Note:** The value reported for CPU load reflects the total usage of the CPU for actual processing performed by the CF, and for host processes other than processes from the CF.

## Procedure

To determine the CPU load on the CFs in a DB2 pureScale instance:

1. Formulate an SQL statement that uses the ENV\_CF\_SYS\_RESOURCES administrative view. For example:

```
SELECT  VARCHAR(NAME,20) AS HOST_ATTRIBUTE,  
        VARCHAR(VALUE,25) AS VALUE,  
        VARCHAR(UNIT,8) AS UNIT  
FROM SYSIBMADM.ENV_CF_SYS_RESOURCES
```

2. Run the statement. The preceding query would return the following output:

| HOST_ATTRIBUTE    | VALUE | UNIT    |
|-------------------|-------|---------|
| HOST_NAME         | HOSTA | -       |
| MEMORY_TOTAL      | 24108 | MB      |
| MEMORY_FREE       | 3504  | MB      |
| MEMORY_SWAP_TOTAL | 4102  | MB      |
| MEMORY_SWAP_FREE  | 4063  | MB      |
| VIRTUAL_MEM_TOTAL | 28211 | MB      |
| VIRTUAL_MEM_FREE  | 7568  | MB      |
| CPU_USAGE_TOTAL   | 96    | PERCENT |
| HOST_NAME         | HOSTB | -       |
| MEMORY_TOTAL      | 24108 | MB      |
| MEMORY_FREE       | 3342  | MB      |
| MEMORY_SWAP_TOTAL | 4102  | MB      |
| MEMORY_SWAP_FREE  | 4063  | MB      |
| VIRTUAL_MEM_TOTAL | 28211 | MB      |
| VIRTUAL_MEM_FREE  | 7406  | MB      |
| CPU_USAGE_TOTAL   | 97    | PERCENT |

16 record(s) selected.

In this output, there are results for both HOSTA and HOSTB, which indicates there are two hosts that are configured to serve as CFs.

3. To determine which of the hosts is acting as the primary CF, you can use the DB2\_CF administrative view:

```
SELECT  VARCHAR(CURRENT_HOST,12) AS HOST,  
        ID,  
        STATE  
FROM SYSIBMADM.DB2_CF
```

The preceding query returns the following output:

| HOST  | ID  | STATE   |
|-------|-----|---------|
| HOSTA | 128 | PRIMARY |
| HOSTB | 129 | PEER    |

2 record(s) selected.

In this case, the primary host is HOSTA, and based on the output from the command used in step 2 on page 1462, you can surmise that the CPU load on the primary CF is 96%.

## What to do next

If you find your CPU usage is running at maximum capacity, adding processors to, or upgrading your cluster caching facilities might improve system throughput.

**Note:** For hosts with more than one logical processor, usage numbers can exceed 100%. For example, an eight processor host might have processor usage approach 800%.

---

## Buffer pool monitoring in a DB2 pureScale environment

Examining the number of times that pages of data requested by a member can be found in group or local buffer pools, as opposed to the number of times they need to be read in from disk can tell you where you might have performance problems related to I/O. Generally speaking, larger buffer pools increase the likelihood that a required page of data can be found in memory.

Viewing and comparing monitor elements related to buffer pool activity can help you understand the extent to which the group buffer pool (GBP) in the cluster caching facility, and the local buffer pools (LBPs) for each member are reducing the amount of disk I/O in your system.

### Monitor elements for viewing DB2 pureScale buffer pool activity

The IBM DB2 pureScale Feature uses a number of monitor elements to report on buffer pool activity across a DB2 pureScale instance.

#### Monitor elements for the group buffer pool

The following monitor elements provide information about the group buffer (GBP) pool in the primary CF:

- “pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element” on page 1044
- “pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element” on page 1045
- “pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element” on page 1042
- “pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element” on page 1075
- “pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements” on page 1077
- “pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element” on page 1074
- “pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element” on page 1132
- “pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element” on page 1134
- “pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element” on page 1131

- “pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element” on page 1024
- “pool\_async\_data\_gbp\_l\_reads - Asynchronous group buffer pool data logical reads monitor element” on page 1025
- “pool\_async\_data\_gbp\_p\_reads - Asynchronous group buffer pool data physical reads monitor element” on page 1025
- “pool\_async\_data\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid data pages monitor element” on page 1024
- “pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element” on page 1029
- “pool\_async\_index\_gbp\_l\_reads - Asynchronous group buffer pool index logical reads monitor element” on page 1030
- “pool\_async\_index\_gbp\_p\_reads - Asynchronous group buffer pool index physical reads monitor element” on page 1030
- “pool\_async\_index\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid index pages monitor element” on page 1029
- “pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element” on page 1035
- “pool\_async\_xda\_gbp\_l\_reads - Group buffer pool XDA data asynchronous logical read requests monitor element” on page 1036
- “pool\_async\_xda\_gbp\_p\_reads - Group buffer pool XDA data asynchronous physical read requests monitor element” on page 1037
- “pool\_async\_xda\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid XDA data pages monitor element” on page 1036

**Note:** These monitor elements report data for the buffer pools for each member individually; no aggregation is performed. If you want to aggregate buffer pool usage information for local buffer pools, such as to calculate the average hit rates across all local buffer pools, use the SUM aggregate function.

Refer to the reference topics for each monitor element to see what monitoring interfaces you can use to examine the data associated with that monitor element.

## Monitor elements for local buffer pools

The following monitor elements provide information about the buffer pools local to each member in the DB2 pureScale instance:

- “pool\_data\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent data pages found in local buffer pool monitor element” on page 1041
- “pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element” on page 1047
- “pool\_index\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool independent index pages found in local buffer pool monitor element” on page 1073
- “pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element” on page 1078

- “pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element” on page 1130
- “pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element” on page 1138
- “pool\_async\_data\_lbp\_pages\_found - Asynchronous local buffer pool data pages found monitor element” on page 1026
- “pool\_async\_index\_lbp\_pages\_found - Asynchronous local buffer pool index pages found monitor element” on page 1031
- “pool\_async\_xda\_lbp\_pages\_found - Asynchronous local buffer pool XDA data pages found monitor element” on page 1037

The values reported for these next three monitor elements represent all reads from disk into the local buffer pool for a given member:

- “pool\_data\_p\_reads - Buffer pool data physical reads monitor element” on page 1050
- “pool\_index\_p\_reads - Buffer pool index physical reads monitor element” on page 1082
- “pool\_xda\_p\_reads - Buffer pool XDA data physical reads monitor element” on page 1139

These last three monitor elements are the same as the ones used in environments *other than* DB2 pureScale environments. The elements are the same, because local buffer pools for a member in a DB2 pureScale environment equivalent to the buffer pools for a database in environments other than DB2 pureScale environments.

**Important:**

- The values reported by these last three monitor elements reflect reads the local buffer pools make from disk of GBP-dependent pages (that is, pages that a member requests from the GBP) that were not found in the GBP. They also include reads from disk of GBP-independent pages (that is, pages that are local to members, and for which members have no dependency on the GBP), such as temporary pages.
- Because of the relationship between LBPs and GBPs in a DB2 pureScale environment, the formulas for calculating buffer pool hit ratios are different than those used outside of a DB2 pureScale environment. See “Formulas for calculating buffer pool hit ratios” on page 1467 for more information.

Refer to the reference topics for each monitor element to see what monitoring interfaces you can use to examine the data associated with that monitor element.

## Buffer pool hit rates and hit ratios in a DB2 pureScale environment

One way of measuring the extent to which pages required by members are found in memory as opposed to on disk is by calculating the buffer pool *hit ratio*. The buffer pool hit ratio indicates the number of times that the database manager found a requested page in a buffer pool (also known as the *hit rate*) as compared to the number of times it had to read it from disk. In a DB2 pureScale environment, both the local buffer pool and group buffer pool hit rates and hit ratios are important factors in assessing overall performance.

Local buffer pool (LBP) hit ratios reflect the extent to which pages that a member needs can be found in a valid state in the local buffer pool. A page in the LBP of a

member is deemed to be in a valid state if that page has not been changed by another member since it was loaded into the LBP. If another member has changed the page, which might happen before the page has been cast out to disk, then the page is said to be *invalid*. If the member with the invalid page requires that page to perform a transaction, the member has to go to the CF to request a new, valid version of the page.

A low LBP hit ratio is an indication that the pages were not found locally, and had to be requested from the CF.

However, in a DB2 pureScale environment, looking at the LBP hit ratios provides only one side of the buffer pool story. You also need to consider the role that the group buffer pool (GBP) plays in retrieving pages, and the hit ratio for the GBP itself. If a member is unable to locate a valid copy of a page in its LBP, it makes a request to the CF to search the GBP for a valid copy of the page. The GBP does one of the following actions:

- If it has a valid copy of the page, the GBP provides it to the member making the request.
- Otherwise, the GBP tells the requesting member that it must read the page from disk.

An additional consideration for LBP usage is the concept of GBP-independent page. A GBP-independent page is a page that is only ever accessed through a LBP of a member, and never exists in the GBP. Pages might be GBP-independent because the operations using the page, or the objects where the pages come from, are only accessed by the local member.

Group buffer pool hit ratios reflect the extent to which pages required by members, for which they do not have a valid local copy, are found in the group buffer pool, as compared to having to be read in from disk. A low hit ratio for the GBP is an indication that relatively few of the pages required by members across the instance are available in the GBP. Increasing the size of the GBP can improve hit rates, and overall performance. Therefore, when calculating the hit ratios for data pages in the local buffer pool (LBP) for a member, you need to consider the number of times the member attempted to read pages from the LBP in comparison to the number of times attempted reads did not find a valid page in the LBP. See “Formulas for calculating buffer pool hit ratios” on page 1467 for details on how LBP and GBP monitor elements are used to calculate the GBP hit rate.

**Tip:** Hit ratios can vary based on many factors, such as the nature of the data in your database, the queries that are run against it, as well as hardware and software configurations. Generally speaking, higher buffer pool hit ratios are reflective of better query performance. If you find hit ratios seem low, or are declining over time, increasing the size of the buffer pools can help. To increase the size of the group buffer pool, adjust the `cf_gbp_sz` configuration parameter on the CF. To adjust local buffer pools, run the **ALTER BUFFERPOOL** statement on the member with the buffer pools that need correction.

## Buffer pool monitor element reporting

In DB2 pureScale environments, as is the case with other DB2 environments, each member reports on its own local buffer pools. No aggregation of data across members takes place. You must take into account which member or members you are interested in, and interpret the data accordingly. In some cases, you might want to calculate the hit ratios for a specific member. In others cases, you might want to

look at the data for all members together, to form an overall view of the hit rates and hit ratios for the DB2 pureScale environment as a whole.

For example, if you submit a query to return data for the number of times a data page was read into a local buffer pool from disk, because it was not found in the GBP (using the `pool_data_gbp_p_reads` monitor element) with the `MON_GET_BUFFERPOOL` table function, and you do not specify which member to return, you will see results like the ones that follow:

```
MEMBER BP_NAME          POOL_DATA_GBP_P_READS
-----
0 IBMDEFAULTBP          408
0 IBMSYSTEMBP4K         0
0 IBMSYSTEMBP8K         0
0 IBMSYSTEMBP16K        0
0 IBMSYSTEMBP32K        0
1 IBMDEFAULTBP          108
1 IBMSYSTEMBP4K         0
1 IBMSYSTEMBP8K         0
1 IBMSYSTEMBP16K        0
1 IBMSYSTEMBP32K        0
2 IBMDEFAULTBP          112
2 IBMSYSTEMBP4K         0
2 IBMSYSTEMBP8K         0
2 IBMSYSTEMBP16K        0
2 IBMSYSTEMBP32K        0
```

15 record(s) selected.

**Important:** In the preceding example, you can see that the data reported for temporary buffer pools shows all zeros. This is not a coincidence; in DB2 pureScale instances, temporary objects and table spaces are local to the member they are associated with. They do not use the GBP on the CF.

If you are interested in the results across all members, you can use the `SUM` aggregate function to add the numbers for all members together:

```
SELECT  VARCHAR(BP_NAME,15) AS BP_NAME,
        SUM(POOL_DATA_GBP_P_READS) AS TOTAL_P_READS
FROM TABLE(MON_GET_BUFFERPOOL('',-2))
GROUP BY BP_NAME
```

The preceding query returns results like the following output:

```
BP_NAME          TOTAL_P_READS
-----
IBMDEFAULTBP          310
IBMSYSTEMBP16K        0
IBMSYSTEMBP32K        0
IBMSYSTEMBP4K         0
IBMSYSTEMBP8K         0
```

5 record(s) selected.

## Formulas for calculating buffer pool hit ratios

Buffer pool hit ratios reflect the extent to which data needed for queries is found in memory, as opposed to having to be read in from external storage. You can calculate hit rates and ratios with formulas that are based on buffer pool monitor elements.

### Local buffer pools

*Table 2126. Formulas for local buffer pool hit ratios.* The formulas shown express the hit ratios as a percentage.

| Type of page | Formula for calculating buffer pool hit ratio   |
|--------------|---|
| Data pages   | $(\text{pool\_data\_lbp\_pages\_found} - \text{pool\_async\_data\_lbp\_pages\_found} - \text{pool\_temp\_data\_l\_reads}) / (\text{pool\_data\_l\_reads}) \times 100$ |

Table 2126. Formulas for local buffer pool hit ratios (continued). The formulas shown express the hit ratios as a percentage.

| Type of page                   | Formula for calculating buffer pool hit ratio   |
|--------------------------------|---|
| Index pages                    | $((\text{pool\_index\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_temp\_index\_l\_reads}) / \text{pool\_index\_l\_reads}) \times 100$   |
| Temporary data pages           | $((\text{pool\_temp\_data\_l\_reads} - \text{pool\_temp\_data\_p\_reads}) / \text{pool\_temp\_data\_l\_reads}) \times 100$  |
| Temporary index pages          | $((\text{pool\_temp\_index\_l\_reads} - \text{pool\_temp\_index\_p\_reads}) / \text{pool\_temp\_index\_l\_reads}) \times 100$   |
| XML storage object (XDA) pages | $((\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}) \times 100$  |
| Overall hit ratio              | $((\text{pool\_data\_lbp\_pages\_found} + \text{pool\_index\_lbp\_pages\_found} + \text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_data\_lbp\_pages\_found} - \text{pool\_async\_index\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / (\text{pool\_data\_l\_reads} + \text{pool\_index\_l\_reads} + \text{pool\_xda\_l\_reads} + \text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads} + \text{pool\_temp\_index\_l\_reads})) \times 100$ |

### Local buffer pools (DB2 pureScale environments)

When calculating LBP hit ratios in a DB2 pureScale environment, the GBP-independent data page hit ratios must also be taken into consideration. When a data page is requested by a member, the LBP is examined first. If the page is found in the LBP, then the **pool\_data\_lbp\_pages\_found** monitor element increments. If the page is not found in the LBP, it is read directly from disk, and the **pool\_data\_p\_reads** monitor element increments. However, the **pool\_data\_lbp\_pages\_found** monitor counter also increments when a GBP-dependent page is found in local buffer pool. The new monitor element **pool\_data\_gbp\_indep\_pages\_found\_in\_lbp** is the only way to distinguish between page access counts in these cases.

Table 2127. Formulas for GBP-independent (LBP) hit ratios. The formulas shown express the hit ratios as a percentage.

| Type of page                   | Formula for calculating buffer pool hit ratio  |
|--------------------------------|--|
| Data pages                     | $(\text{pool\_data\_gbp\_indep\_pages\_found\_in\_lbp} / (\text{pool\_data\_gbp\_indep\_pages\_found\_in\_lbp} + \text{pool\_data\_p\_reads} + \text{pool\_temp\_data\_p\_reads} - \text{pool\_data\_gbp\_p\_reads})) \times 100$      |
| Index pages                    | $(\text{pool\_index\_gbp\_indep\_pages\_found\_in\_lbp} / (\text{pool\_index\_gbp\_indep\_pages\_found\_in\_lbp} + \text{pool\_index\_p\_reads} + \text{pool\_temp\_index\_p\_reads} - \text{pool\_index\_gbp\_p\_reads})) \times 100$ |
| XML storage object (XDA) pages | $(\text{pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp} / (\text{pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp} + \text{pool\_xda\_p\_reads} + \text{pool\_temp\_xda\_p\_reads} - \text{pool\_xda\_gbp\_p\_reads})) \times 100$           |

For GBP-independent pages, tuning the LBP size will affect the hit ratio. For certain type of operations which are independent of the GBP, like temporary page access or NOT LOGGED INITIALLY operations, you will want to monitor the GBP-independent page hit ratio.

Alternatively, to calculate the local buffer pool hit ratio for a GBP-dependent page agent, you can use the following formulas.

Table 2128. Formulas for GBP-dependent (LBP) hit ratios. The formulas shown express the hit ratios as a percentage.

| Type of page                   | Formula for calculating buffer pool hit ratio  |
|--------------------------------|--|
| Data pages                     | $((\text{pool\_data\_lbp\_pages\_found} - \text{pool\_data\_gbp\_indep\_pages\_found\_in\_lbp}) / (\text{pool\_data\_l\_reads} - \text{pool\_data\_gbp\_indep\_pages\_found\_in\_lbp} - (\text{pool\_data\_p\_reads} - \text{pool\_data\_gbp\_p\_reads}))) \times 100$       |
| Index pages                    | $((\text{pool\_index\_lbp\_pages\_found} - \text{pool\_index\_gbp\_indep\_pages\_found\_in\_lbp}) / (\text{pool\_index\_l\_reads} - \text{pool\_index\_gbp\_indep\_pages\_found\_in\_lbp} - (\text{pool\_index\_p\_reads} - \text{pool\_index\_gbp\_p\_reads}))) \times 100$ |
| XML storage object (XDA) pages | $((\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp}) / (\text{pool\_xda\_l\_reads} - \text{pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp} - (\text{pool\_xda\_p\_reads} - \text{pool\_xda\_gbp\_p\_reads}))) \times 100$             |

You will need to compare both the LBP hit ratio and the GBP hit ratio to decide how to tune both buffer pools, or to validate the results after tuning the GBP and the LBPs.

### Group buffer pools (DB2 pureScale environments)

The formulas used to calculate group buffer pool hit ratios in a DB2 pureScale environment are different from formulas for hit ratios used in other DB2 environments. This difference is because of how the group buffer pool in the cluster caching facility works with the local buffer pools in each member to retrieve pages of data. The following formulas, which are based on buffer pool monitor elements, can be used to calculate hit ratios for data, index, and XML storage object pages, for both the local and group buffer pools.

Table 2129. Formulas for group buffer pool (GBP) hit ratios. The formulas shown express the hit ratios as a percentage.

| Type of page                   | Formula for calculating buffer pool hit ratio   |
|--------------------------------|---|
| Data pages                     | $((\text{pool\_data\_gbp\_l\_reads} - \text{pool\_data\_gbp\_p\_reads}) / \text{pool\_data\_gbp\_l\_reads}) \times 100$   |
| Index pages                    | $((\text{pool\_index\_gbp\_l\_reads} - \text{pool\_index\_gbp\_p\_reads}) / \text{pool\_index\_gbp\_l\_reads}) \times 100$  |
| XML storage object (XDA) pages | $((\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}) \times 100$  |
| Overall hit ratio              | $((\text{pool\_data\_gbp\_l\_reads} + \text{pool\_index\_gbp\_l\_reads} + \text{pool\_xda\_gbp\_l\_reads} - \text{pool\_data\_gbp\_p\_reads} - \text{pool\_index\_gbp\_p\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / (\text{pool\_data\_gbp\_l\_reads} + \text{pool\_index\_gbp\_l\_reads} + \text{pool\_xda\_gbp\_l\_reads})) \times 100$ |

In addition to the preceding formulas for calculating buffer pool hit ratios, you can also use the following formulas to show what percentage of the time pages that are prefetched are found in the GBP:

#### Prefetches for data pages

$$((\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}) / \text{pool\_async\_data\_gbp\_l\_reads}) \times 100$$

#### Prefetches for index pages

$$((\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}) / \text{pool\_async\_index\_gbp\_l\_reads}) \times 100$$

#### Prefetches for XML storage object (XDA) pages

$$((\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}) / \text{pool\_async\_xda\_gbp\_l\_reads}) \times 100$$

## Calculating buffer pool hit ratios in a DB2 pureScale environment

Calculating buffer pool hit ratios for a DB2 pureScale instance can help you understand where there are opportunities to tune buffer pools to improve I/O efficiency.

### Before you begin

Determine which ratio or ratios you are interested in. If you want to see a ratio across all members in an instance, consider formulating your SQL to aggregate data across members using the SUM aggregate function. If you are interested in seeing the data for a specific member only, you can use specify the member for which you want to see data in the MON\_GET\_BUFFERPOOL table function.

### Procedure

To calculate buffer pool hit ratios, follow these steps:

1. Retrieve the information for the required monitor elements. This example uses the MON\_GET\_BUFFERPOOL table function to retrieve the monitor elements that contain the values needed to calculate the hit ratio for data pages for the GBP, **pool\_data\_gbp\_l\_reads** and **pool\_data\_gbp\_p\_reads**.

```
SELECT varchar(bp_name,20) AS bp_name,
       pool_data_gbp_l_reads,
       pool_data_gbp_p_reads,
       member
FROM TABLE(MON_GET_BUFFERPOOL(' ', -2))
```

The preceding query returns data like the following example:

| BP_NAME        | POOL_DATA_GBP_L_READS | POOL_DATA_GBP_P_READS | MEMBER |
|----------------|-----------------------|-----------------------|--------|
| IBMDEFAULTBP   | 1814911               | 456990                | 1      |
| IBMSYSTEMBP4K  | 0                     | 0                     | 1      |
| IBMSYSTEMBP8K  | 0                     | 0                     | 1      |
| IBMSYSTEMBP16K | 0                     | 0                     | 1      |
| IBMSYSTEMBP32K | 0                     | 0                     | 1      |
| IBMDEFAULTBP   | 1807959               | 455287                | 3      |
| IBMSYSTEMBP4K  | 0                     | 0                     | 3      |
| IBMSYSTEMBP8K  | 0                     | 0                     | 3      |
| IBMSYSTEMBP16K | 0                     | 0                     | 3      |
| IBMSYSTEMBP32K | 0                     | 0                     | 3      |
| IBMDEFAULTBP   | 1813932               | 455225                | 2      |
| IBMSYSTEMBP4K  | 0                     | 0                     | 2      |
| IBMSYSTEMBP8K  | 0                     | 0                     | 2      |
| IBMSYSTEMBP16K | 0                     | 0                     | 2      |
| IBMSYSTEMBP32K | 0                     | 0                     | 2      |
| IBMDEFAULTBP   | 1113396               | 278845                | 0      |
| IBMSYSTEMBP4K  | 0                     | 0                     | 0      |
| IBMSYSTEMBP8K  | 0                     | 0                     | 0      |
| IBMSYSTEMBP16K | 0                     | 0                     | 0      |
| IBMSYSTEMBP32K | 0                     | 0                     | 0      |

20 record(s) selected.

**Important:** In the preceding example, you can see that the data reported for temporary buffer pools shows all zeros. This is not a coincidence; in DB2 pureScale instances, temporary objects and table spaces are local to the member they are associated with. They do not use the GBP on the CF.

2. Use the values returned for the monitor elements to calculate the hit ratio. The formula for calculating the hit ratio for the GBP (expressed as a percentage) is

$$((\text{pool\_data\_gbp\_l\_reads} - \text{pool\_data\_gbp\_p\_reads}) \div \text{pool\_data\_gbp\_l\_reads}) \times 100$$

So, using the data returned for the monitor elements in step 1:

$$\begin{aligned}
&(((1,814,911+1,807,959 + 1,813,932+1,113,396) - (456,990+455,287 + 455,225+278,845)) \div (1,814,911+1,807,959 + 1,813,932+1,113,396)) \times 100 \\
&= ((6,550,198 - 1,646,347) \div 6,550,198) \times 100 \\
&= 74.9\%
\end{aligned}$$

In this example, the hit ratio for the GBP is 74.9%

**Note:** The values shown in the output for queries are for illustrative purposes only.

## Example

*Example 1: Find the overall hit rates across all members*

This example is similar to the one shown in the preceding procedure, except that it uses an aggregate function to provide overall hit rates across all members.

```

SELECT VARCHAR(BP_NAME,20) AS BP,
       SUM(POOL_DATA_GBP_L_READS) AS POOL_DATA_GBP_L_READS,
       SUM(POOL_DATA_GBP_P_READS) AS POOL_DATA_GBP_P_READS
FROM TABLE(MON_GET_BUFFERPOOL('',-2))
GROUP BY BP_NAME

```

Results:

| BP             | POOL_DATA_GBP_L_READS | POOL_DATA_GBP_P_READS |
|----------------|-----------------------|-----------------------|
| IBMDEFAULTBP   | 6550198               | 1646347               |
| IBMSYSTEMBP16K | 0                     | 0                     |
| IBMSYSTEMBP32K | 0                     | 0                     |
| IBMSYSTEMBP4K  | 0                     | 0                     |
| IBMSYSTEMBP8K  | 0                     | 0                     |

5 record(s) selected.

*Example 2: Determining the GBP hit ratio for all data, index, and XML storage object (XDA) pages*

This example uses the **MON\_GET\_BUFFERPOOL** table function to retrieve the data contained in the required monitor elements, and calculates the hit ratio for each member. To calculate the GBP hit ratio for all data, index, and XDA pages, use the following formula:

$$\begin{aligned}
&((pool\_data\_gbp\_l\_reads + pool\_index\_gbp\_l\_reads+pool\_xda\_gbp\_l\_reads) \\
&- (pool\_data\_gbp\_p\_reads + pool\_index\_gbp\_p\_reads+pool\_xda\_gbp\_p\_reads )) \\
&\div (pool\_data\_gbp\_l\_reads + pool\_index\_gbp\_l\_reads+pool\_xda\_gbp\_l\_reads) \times 100
\end{aligned}$$

```

WITH BPMETRICS AS (
  SELECT BP_NAME,
         POOL_DATA_GBP_L_READS +
         POOL_INDEX_GBP_L_READS +
         POOL_XDA_GBP_L_READS
         AS LOGICAL_READS,
         POOL_DATA_GBP_P_READS +
         POOL_INDEX_GBP_P_READS +
         POOL_XDA_GBP_P_READS
         AS PHYSICAL_READS,
         MEMBER
  FROM TABLE(MON_GET_BUFFERPOOL('',-2)) AS METRICS)
SELECT VARCHAR(BP_NAME,20) AS BP_NAME,
       LOGICAL_READS,
       PHYSICAL_READS,
       CASE WHEN LOGICAL_READS > 0
            THEN DEC(((
              FLOAT(LOGICAL_READS) - FLOAT(PHYSICAL_READS)) /
              FLOAT(LOGICAL_READS))
              * 100,5,2)
            ELSE NULL END AS HIT_RATIO,
       MEMBER
FROM BPMETRICS

```

Results:

| BP_NAME        | LOGICAL_READS | PHYSICAL_READS | HIT_RATIO | MEMBER |
|----------------|---------------|----------------|-----------|--------|
| IBMDEFAULTBP   | 5730213       | 617628         | 89.22     | 1      |
| IBMSYSTEMBP4K  | 0             | 0              | -         | 1      |
| IBMSYSTEMBP8K  | 0             | 0              | -         | 1      |
| IBMSYSTEMBP16K | 0             | 0              | -         | 1      |
| IBMSYSTEMBP32K | 0             | 0              | -         | 1      |
| IBMDEFAULTBP   | 5724845       | 615395         | 89.25     | 3      |
| IBMSYSTEMBP4K  | 0             | 0              | -         | 3      |
| IBMSYSTEMBP8K  | 0             | 0              | -         | 3      |
| IBMSYSTEMBP16K | 0             | 0              | -         | 3      |
| IBMSYSTEMBP32K | 0             | 0              | -         | 3      |
| IBMDEFAULTBP   | 5731714       | 615814         | 89.25     | 2      |
| IBMSYSTEMBP4K  | 0             | 0              | -         | 2      |
| IBMSYSTEMBP8K  | 0             | 0              | -         | 2      |
| IBMSYSTEMBP16K | 0             | 0              | -         | 2      |
| IBMSYSTEMBP32K | 0             | 0              | -         | 2      |
| IBMDEFAULTBP   | 5024809       | 409159         | 91.85     | 0      |
| IBMSYSTEMBP4K  | 0             | 0              | -         | 0      |
| IBMSYSTEMBP8K  | 0             | 0              | -         | 0      |
| IBMSYSTEMBP16K | 0             | 0              | -         | 0      |
| IBMSYSTEMBP32K | 0             | 0              | -         | 0      |

20 record(s) selected.

*Example 3: Using the SUM aggregate function to compute an overall hit ratio*

You can also use the SUM aggregate function to compute an overall hit ratio across all members as follows:

```

WITH BPMETRICS AS (
  SELECT SUM(PPOOL_DATA_GBP_L_READS) +
         SUM(PPOOL_INDEX_GBP_L_READS) +
         SUM(PPOOL_XDA_GBP_L_READS)
  AS LOGICAL_READS,
  SUM(PPOOL_DATA_GBP_P_READS) +
  SUM(PPOOL_INDEX_GBP_P_READS) +
  SUM(PPOOL_XDA_GBP_P_READS)
  AS PHYSICAL_READS
  FROM TABLE(MON_GET_BUFFERPOOL(' ', -2)) AS METRICS)
SELECT LOGICAL_READS,
       PHYSICAL_READS,
       CASE WHEN LOGICAL_READS > 0
       THEN DEC(((FLOAT(LOGICAL_READS) - FLOAT(PHYSICAL_READS)) /
                FLOAT(LOGICAL_READS))
                * 100, 5, 2)
       ELSE NULL END AS HIT_RATIO
FROM BPMETRICS

```

Results:

| LOGICAL_READS | PHYSICAL_READS | HIT_RATIO |
|---------------|----------------|-----------|
| 22211581      | 2255996        | 89.84     |

1 record(s) selected.

**What to do next**

If hit ratios seem low, or if they decline over time, you might want to increase the size of the buffer pools on either members, CFs, or both. If you are seeing lower than expected hit rates for the LBPs overall across the DB2 pureScale instance, look at the hit rates for each member individually, since the buffer pools on each member can have different sizes. An smaller sized LBP on one member might be unduly influencing the average hit rate for the instance.

**Tip:** Hit ratios can vary based on many factors, such as the nature of the data in your database, the queries that are run against it, as well as hardware and software configurations. Generally speaking, higher buffer pool hit ratios are reflective of better query performance. If you find hit ratios seem low, or are declining over time, increasing the size of the buffer pools can help. To increase the size of the

group buffer pool, adjust the `cf_gbp_sz` configuration parameter on the CF. To adjust local buffer pools, run the `ALTER BUFFERPOOL` statement on the member with the buffer pools that need correction.

---

## Lock monitoring in a DB2 pureScale environment overview

As with traditional DB2 environments, lock management in a DB2 pureScale environment is essential for maintaining both data integrity and high levels of concurrency. Locking across members in a DB2 pureScale environment is managed by the global lock manager (GLM) component of the cluster caching facility. Monitoring locking in a DB2 pureScale environment involves reviewing not only locks that might be held within a member, but also lock waits between members.

In a DB2 pureScale environment, the fact that different members work with the same data introduces the possibility of another type of contention for data: when two members want to update the same object. When a member needs a lock for an object, the local lock manager (LLM) component within the member works with the global lock manager (GLM): if the LLM does not already hold a lock for the object in question, the LLM requests a lock from the GLM. In this way, the GLM mediates requests for locks from different members.

When viewed at the global level for the DB2 pureScale instance, monitor elements such as `locks_held`, or `lock_wait_time` report data on *all* locks in the instance, both within and between members. Monitor elements added specifically for the DB2 pureScale Feature can be used to examine just the lock waits between members.

### Lock requests between members

In a DB2 pureScale environment, an application on one member might request a lock for an object currently locked by another member. The DB2 pureScale Feature introduces monitor elements that specifically report information about locks across members.

As is the case in traditional DB2 environments, processing *within* members in a DB2 pureScale environment can result in locking of objects as one application attempts to perform an operation that is not compatible with an operation being performed by another application. This can occur, for example, if two applications attempt to update the same row of data at the same time. You can monitor the extent to which this type of locking within a member takes place using the locking event monitor to view lock-related information. In a DB2 pureScale environment, lock waits might also occur *between* members, as one member requests a lock for an object currently locked by an application on a different member. So, in a DB2 pureScale environment, in addition to examining locking on individual members, you might also want to look at cross-member lock information.

### Lock waits between members

The following monitor elements report *only* the portion of time that an application has been awaiting a lock held by another member:

- `lock_wait_time_global`
- `lock_wait_time_global_top`
- `lock_waits_global`
- `lock_timeouts_global`

The time reported by these monitor elements is included as part of the overall **lock\_wait\_time** and **lock\_wait\_time\_top** monitor elements when these elements are viewed from the perspective of the DB2 pureScale instance as a whole. Similarly, the count of lock waits and time-outs is reported as part of both the **lock\_waits** and **lock\_timeouts** monitor elements when these monitor elements are viewed at the level of the instance as a whole.

The following scenario illustrates the relationship between these cross-member, or global lock-wait monitor elements and the monitor elements that report locks within a member:

1. Application 1 on member 1 holds a shared (S) lock on a row.
2. Application 2 on member 2 requests an exclusive (X) lock on the same row. Application 2 is forced to wait, as the row is currently locked by application 1.
3. 2 ms later, application 3 on member 2 requests an exclusive (X) lock on the same row. Application 3 is also forced to wait.
4. 8 ms later, application 1 releases its lock, and application 2 acquires its lock.
5. 5 ms later, application 2 releases its lock, and application 3 acquires its lock.

The total time spent waiting for locks, as reported by **lock\_wait\_time** is 23 ms; application 2 had to wait for 10 ms in total, while application 3 had to wait for 13 ms in total. However, the amount of time spent waiting for a lock *between* members, **lock\_wait\_time\_global** is only 10 ms, as this wait time is the only portion of the overall lock wait time that involved one member waiting on a lock held by another.

Similarly, the count of the number of locks held reported by **lock\_waits\_global** is 1. The wait by application 2 on application 1 counts as one member waiting on another. Even though application 3 was, for a portion of the time it was waiting, held up by an application on member 1, this lock wait is not counted as a lock wait between members because it obtained its lock from the local lock manager on member 2.

## Reporting of applications holding locks

The locking event monitor shows information about applications holding locks. Generally speaking, the locking event monitor reports application information regardless of which member the application is running on. However, in rare cases, it might not be possible to determine which application is holding a lock when that application is running on a remote member. Consider the following example:

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES
FOR LOCKS WAITING MORE THAN 5 SECONDS
```

This statement causes locks held longer than 5 seconds for the “finance” workload to be recorded in a lock event monitor. Now consider the following scenario.

- A lock is held by application 1 on member 1.
- After 5 seconds, the system attempts to write information about the lock to the lock event monitor, including application and member information. However, if the lock is released by application 1 and then a lock is immediately granted for the same object to another application on another member, the application holding the lock might be recorded as “Lock holder was unavailable for collection”.

Also note that if a member that held a lock fails, the locking event monitor is not able to report information about which applications running on that member might have held a lock on a required piece of data.

Finally, there are some limitations on reporting applications holding locks when you use deprecated monitoring features. See “Monitoring locking with snapshot monitor” on page 1481 for more information.

## Monitor elements for viewing locks between members

The IBM DB2 pureScale Feature adds a number of new monitor elements that you can use to monitor locking in a DB2 pureScale environment. These monitor elements report specifically on information about lock waits between members in a DB2 pureScale instance.

### Locking-related monitor elements

The following monitor elements can be used to get information about lock waits between members and lock escalations:

- “lock\_waits\_global - Lock waits global monitor element” on page 930
- “lock\_wait\_time\_global - Lock wait time global monitor element” on page 926
- “lock\_wait\_time\_global\_top - Top global lock wait time monitor element” on page 928
- “lock\_timeouts\_global - Lock timeouts global monitor element” on page 922
- “lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element” on page 910
- “lock\_escals\_locklist - Number of locklist lock escalations monitor element” on page 909
- “lock\_escals\_global - Number of global lock escalations monitor element” on page 907

While not directly related to locking, the following elements can be used to get information about the extent to which things like statements, units of work or workloads in a DB2 pureScale instance wait on a cluster caching facility:

- “cf\_waits - Number of cluster caching facility waits monitor element” on page 703
- “cf\_wait\_time - cluster caching facility wait time monitor element” on page 703

The cf\_wait\_time element shows how long a statement, unit or work, or workload has had to wait for the cluster caching facility to service a request. Use this element in conjunction with cf\_waits ( $cf\_wait\_time \div cf\_waits$ ) to find the average wait time for each request. Generally speaking, the average wait time for requests to be serviced by the cluster caching facility is on the order of a few milliseconds. If you find they are substantially longer (that is, an order of magnitude or more), you might have an InfiniBand setup issue. Alternatively, your cluster caching facility may be overwhelmed with requests that it is not able to keep up with.

Refer to the reference topics for each monitor element to see what monitoring interfaces you can use to examine the data associated with that monitor element.

You can also use the other monitor elements supported by the locking event monitor to view information about locking *within* members.

---

## Page reclaiming

Different members in a DB2 pureScale instance might require access to a page of data that another member is already using. The process whereby one member requests and is granted a page being used by another member is known as *page reclaiming*.

If different members require access to the same page of data, the cluster caching facility manages which member accesses the page and when. In some cases, the cluster caching facility (also known as the CF) might permit one member to *reclaim* the page from another member before the other member is finished with it. The following example illustrates how page reclaiming occurs:

Assume there are two members, M1 and M2 that intend to update two different rows on the same page of data.

1. M1 is doing an update on a row R1 within a page of data. It is granted exclusive access to the page containing that row of data.
2. M2 requires an exclusive to the same page to update row R2. It passes this request to the CF. M2 waits while the request is processed.
3. The CF sees that member M1 already has an exclusive access to the page. It issues a request to M1 to reclaim the page. In the meantime, M2 waits.
4. M1 processes the reclaim request by writing the page back to the GBP and then releasing the page. (M1 retains any row or table locks it might have.)
5. The CF grants M2 access to the page. M2 reads the page from the GBP to perform whatever operations for which the page is needed.

It is important to note that as pointed out in step 4, any locks that a member has on rows or tables for the purposes of updates are retained until the unit of work has completed, even if another member reclaims and begins using the page before the end of that unit of work. In this way, different members can work with the same page of data without compromising lock integrity. If it happens that two members require incompatible row locks to the same row of data, then as is the case with lock management on a single member, one member must complete its processing before the second is allowed to proceed.

## Monitor elements for page reclaiming

The IBM DB2 pureScale Feature adds a number of new monitor elements that you can use to monitor the extent to which page reclaiming is taking place in a DB2 pureScale instance.

### Page reclaiming-related monitor elements

The following monitor elements can be used to get information about the extent to which page reclaiming is taking place in a DB2 pureScale instance:

- “page\_reclaims\_x - Page reclaims exclusive access monitor element” on page 1010
- “page\_reclaims\_initiated\_s - Page reclaims initiated shared access monitor element” on page 1011
- “spacemappage\_page\_reclaims\_x - Space map page reclaims exclusive access monitor element” on page 1234
- “spacemappage\_page\_reclaims\_s - Space map page reclaims shared access monitor element” on page 1234

- “page\_reclaims\_initiated\_x - Page reclaims initiated exclusive access monitor element” on page 1011
- “page\_reclaims\_initiated\_s - Page reclaims initiated shared access monitor element” on page 1011
- “spacemappage\_page\_reclaims\_initiated\_x - Space map page reclaims initiated exclusive access monitor element” on page 1235
- “spacemappage\_page\_reclaims\_initiated\_s - Space map page reclaims initiated shared access monitor element” on page 1235
- “reclaim\_wait\_time - Reclaim wait time monitor element” on page 1170
- “spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element” on page 1236

Refer to the reference topics for each monitor element to see what monitoring interfaces you can use to examine the data associated with that monitor element.

## Monitoring page reclaiming between members

When examining where a particular application or statement is spending its time, in addition to the time spent waiting for locks, applications, or statements in a DB2 pureScale environment might need to wait for a page to become available when it is in use by another member. You can use page reclaiming monitor elements to view the extent to which this type of wait might be affecting throughput on your system.

### About this task

To view page reclaiming statistics, use the MON\_GET\_PAGE\_ACCESS\_INFO table function. This table function returns object-level information about the extent to which members both request pages currently in use by other members, and the extent to which members release those pages at the request of other members. You can also retrieve the wait times involved.

### Procedure

1. Determine what types of pages you are interested in viewing results for. The example that follows retrieves information about the number of times pages were reclaimed for all data and index pages, using the page\_reclaims\_x and page\_reclaims\_s monitor elements.
2. Formulate an SQL statement that uses the MON\_GET\_PAGE\_ACCESS\_INFO table function. For example, to retrieve information about data and index pages reclaimed for all members, you can construct a statement like the one that follows:

```
SELECT MEMBER,
       VARCHAR(TABNAME,30) AS TABLE,
       VARCHAR(OBJTYPE,8) AS OBJTYPE,
       PAGE_RECLAIMS_X,
       PAGE_RECLAIMS_S
FROM TABLE(MON_GET_PAGE_ACCESS_INFO('DTW',' ', -2))
WHERE PAGE_RECLAIMS_X !=0 OR PAGE_RECLAIMS_S !=0
ORDER BY MEMBER ASC, PAGE_RECLAIMS_X ASC
```

3. Run the query. In this case, the results returned would look like the following example:

| MEMBER | TABLE           | OBJTYPE | PAGE_RECLAIMS_X | PAGE_RECLAIMS_S |
|--------|-----------------|---------|-----------------|-----------------|
| 0      | CUSTOMER        | TABLE   | 196             | 0               |
| 0      | STOCK_1_250     | TABLE   | 213             | 0               |
| 0      | STOCK_1251_1500 | TABLE   | 237             | 0               |
| 0      | STOCK_251_500   | TABLE   | 239             | 0               |

|   |                 |       |        |      |
|---|-----------------|-------|--------|------|
| 0 | STOCK_501_750   | TABLE | 245    | 0    |
| 0 | STOCK_1751_2000 | TABLE | 253    | 0    |
| 0 | STOCK_2001_2250 | TABLE | 254    | 0    |
| 0 | STOCK_751_1000  | TABLE | 259    | 0    |
| 0 | STOCK_1501_1750 | TABLE | 269    | 0    |
| 0 | STOCK_2251_2500 | TABLE | 274    | 0    |
| 0 | STOCK_251_500   | INDEX | 276    | 2934 |
| 0 | STOCK_1001_1250 | TABLE | 280    | 0    |
| 0 | STOCK_1501_1750 | INDEX | 284    | 3070 |
| 0 | STOCK_501_750   | INDEX | 294    | 3029 |
| 0 | STOCK_1_250     | INDEX | 296    | 2916 |
| 0 | STOCK_751_1000  | INDEX | 301    | 3056 |
| 1 | STOCK_1001_1250 | TABLE | 247    | 0    |
| 1 | STOCK_501_750   | TABLE | 255    | 0    |
| 1 | STOCK_751_1000  | TABLE | 257    | 0    |
| 1 | STOCK_1501_1750 | TABLE | 257    | 0    |
| 1 | STOCK_251_500   | INDEX | 287    | 2921 |
| 1 | STOCK_1_250     | INDEX | 292    | 2916 |
| 1 | STOCK_751_1000  | INDEX | 316    | 3190 |
| 1 | STOCK_501_750   | INDEX | 319    | 2956 |
| 1 | ORDERS          | INDEX | 42434  | 1416 |
| 1 | ORDER_LINE      | INDEX | 116107 | 3731 |
| 2 | CUSTOMER        | TABLE | 180    | 0    |
| 2 | STOCK_2001_2250 | TABLE | 221    | 0    |
| . | .               | .     | .      | .    |
| . | .               | .     | .      | .    |
| 2 | STOCK_1501_1750 | TABLE | 240    | 0    |
| 2 | STOCK_2251_2500 | TABLE | 247    | 0    |
| 2 | STOCK_1251_1500 | TABLE | 268    | 0    |
| 2 | STOCK_251_500   | INDEX | 276    | 2976 |
| 2 | STOCK_1_250     | INDEX | 284    | 2846 |
| 2 | STOCK_501_750   | TABLE | 285    | 0    |
| 2 | STOCK_501_750   | INDEX | 293    | 3143 |
| 2 | DISTRICT        | TABLE | 18402  | 0    |
| 2 | ORDERS          | INDEX | 41581  | 1474 |
| 2 | ORDER_LINE      | INDEX | 114442 | 3815 |
| 3 | CUSTOMER        | TABLE | 159    | 0    |
| 3 | STOCK_251_500   | TABLE | 226    | 0    |
| 3 | ORDERS          | INDEX | 42192  | 1340 |
| 3 | ORDER_LINE      | INDEX | 115459 | 3871 |

112 record(s) selected.

**Note:** Part of the lengthy output from the query has been excluded, as denoted by the vertical ellipsis.

## Results

In the preceding example, you can see that information for data and index pages is returned separately. Also, the schema is specified to exclude data from objects associated with the SYSIBM schema from being reported.

## Example

*Example 1: Retrieving page reclaim wait times*

The following SQL retrieves the total pages reclaimed and total wait time across all members.

```
SELECT
    SUM(PAGE_RECLAIMS_X+PAGE_RECLAIMS_S+SPACEMAPPAGE_PAGE_RECLAIMS_X
        +SPACEMAPPAGE_PAGE_RECLAIMS_S)AS PAGE_RECLAIMS,
    SUM(RECLAIM_WAIT_TIME) AS RECLAIM_WAIT_TIME
FROM TABLE(MON_GET_PAGE_ACCESS_INFO('',' ', -2))
```

The results of this query would look like the following example::

| PAGE_RECLAIMS | RECLAIM_WAIT_TIME |
|---------------|-------------------|
| 156           | 91                |

1 record(s) selected.

(Wait time is reported in milliseconds)

Example 2: Show the 10 tables that are associated with the highest number of pages reclaimed

This example shows how you can see which table objects are involved with page reclaiming.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA,
       SUBSTR(TABNAME,1,20) AS TABNAME,
       RECLAIM_WAIT_TIME,
       MEMBER,
       SUBSTR(OBJTYPE,1,10) AS OBJTYPE
FROM TABLE(MON_GET_PAGE_ACCESS_INFO(NULL,NULL,-2))
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY RECLAIM_WAIT_TIME DESC
FETCH FIRST 10 ROWS ONLY
```

Results:

| TABSCHEMA | TABNAME    | RECLAIM_WAIT_TIME | MEMBER | OBJTYPE |
|-----------|------------|-------------------|--------|---------|
| DTW       | ORDER_LINE | 1307192           | 1      | INDEX   |
| DTW       | ORDER_LINE | 1250134           | 2      | INDEX   |
| DTW       | ORDER_LINE | 1249452           | 0      | INDEX   |
| DTW       | ORDER_LINE | 1159741           | 3      | INDEX   |
| DTW       | DISTRICT   | 827598            | 0      | TABLE   |
| DTW       | DISTRICT   | 785354            | 2      | TABLE   |
| DTW       | DISTRICT   | 767148            | 1      | TABLE   |
| DTW       | DISTRICT   | 687608            | 3      | TABLE   |
| DTW       | ORDERS     | 556538            | 0      | INDEX   |
| DTW       | ORDERS     | 539858            | 2      | INDEX   |

10 record(s) selected.

(Wait time is reported in milliseconds)

Example 3: Show the 10 statements that are causing the highest number of pages reclaimed

This query is a variation on the preceding example; in this case, the query returns the 10 statements associated with the highest number of pages reclaimed:

```
SELECT SUBSTR(STMT_TEXT,1,50) AS STMT_TEXT,
       RECLAIM_WAIT_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY RECLAIM_WAIT_TIME DESC
FETCH FIRST 10 ROWS ONLY
```

Results:

| STMT_TEXT  | RECLAIM_WAIT_TIME |
|--|-------------------|
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03 | 796668            |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03 | 785863            |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03 | 746521            |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03 | 623461            |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ? | 610602            |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ? | 522899            |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ? | 518076            |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = | 419022            |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = | 406028            |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = | 406006            |

10 record(s) selected.

(Wait time is reported in milliseconds)

Example 4: Show the 10 statements that are causing the highest number of pages reclaimed, with the average wait time for each execution of each statement

In the preceding example, the wait time is expressed as an overall value per statement. The query does not take into account the fact that a given statement might have run numerous times. This example shows how you can examine the average wait time for each execution of each of the top 10 statements:

```
SELECT SUBSTR(STMT_TEXT,1,75) AS STMT_TEXT,
       NUM_EXECUTIONS,
       RECLAIM_WAIT_TIME,
       DEC(FLOAT(RECLAIM_WAIT_TIME)/FLOAT(NUM_EXECUTIONS),10,8)
       AS AVG_WAIT_PEREXEC
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY AVG_WAIT_PEREXEC DESC
FETCH FIRST 10 ROWS ONLY
```

Results:

| STMT_TEXT   | NUM_EXECUTIONS | RECLAIM_WAIT_TIME | AVG_WAIT_PEREXEC |
|---|----------------|-------------------|------------------|
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?       | 157173         | 419497            | 2.66901439       |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?       | 155752         | 397870            | 2.55450973       |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?       | 155352         | 385613            | 2.48218883       |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?       | 155151         | 347847            | 2.24199006       |
| Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?)                         | 157173         | 259076            | 1.64834927       |
| Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?)                         | 155752         | 253548            | 1.62789562       |
| Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?)                         | 155352         | 232300            | 1.49531386       |
| Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?)                         | 155151         | 219607            | 1.41544044       |
| Delete from NEW_ORDER where NO_W_ID = ? and NO_D_ID = ? and NO_O_ID = ? | 152968         | 106525            | 0.69638747       |
| Delete from NEW_ORDER where NO_W_ID = ? and NO_D_ID = ? and NO_O_ID = ? | 152591         | 101367            | 0.66430523       |

10 record(s) selected.

(Wait time is reported in milliseconds)

A slightly different version of this query shows how long each statement took to execute:

```
SELECT SUBSTR(STMT_TEXT,1,75) AS STMT_TEXT,
       NUM_EXECUTIONS,
       RECLAIM_WAIT_TIME,
       DEC(FLOAT(RECLAIM_WAIT_TIME)/FLOAT(NUM_EXECUTIONS),10,8) AS AVG_EXEC_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY RECLAIM_WAIT_TIME DESC
FETCH FIRST 10 ROWS ONLY
```

Results:

| STMT_TEXT   | NUM_EXECUTIONS | RECLAIM_WAIT_TIME | AVG_EXEC_TIME |
|---|----------------|-------------------|---------------|
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S | 1555470        | 755544            | 0.48573357    |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S | 1554405        | 754231            | 0.48522167    |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S | 1570256        | 741047            | 0.47192750    |
| Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S | 1550835        | 707148            | 0.45597887    |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?)                      | 1554392        | 508568            | 0.32718130    |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?)                      | 1555454        | 497197            | 0.31964751    |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?)                      | 1570245        | 493692            | 0.31440444    |
| Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?)                      | 1550813        | 465049            | 0.29987432    |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?           | 157145         | 419283            | 2.66812816    |
| Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?           | 155719         | 397364            | 2.55180164    |

10 record(s) selected.

(Wait time is reported in milliseconds)

---

## Chapter 14. Using deprecated monitoring features in a DB2 pureScale environment

The IBM DB2 pureScale Feature extends the DB2 monitoring infrastructure with a rich set of monitor elements that you can use to retrieve information about a DB2 pureScale instance. However, there are some limitations to be aware of when retrieving and interpreting monitoring data using deprecated monitoring interfaces.

This topic describes the limitations to keep in mind when you are using any of the following deprecated features:

- “Monitoring locking with snapshot monitor”
- “Deadlock event monitor” on page 1482
- “LIST TABLESPACES and LIST TABLESPACE CONTAINERS commands” on page 1485

### Monitoring locking with snapshot monitor

If you use snapshot monitoring commands, functions or views to examine information about locks between members, details about the applications holding locks are only displayed if the application is running on the same member as where the snapshot is being taken. Otherwise, the ID of the application holding the lock are reported as REMOTE APPLICATION, and other information, such as the application ID and the lock mode are omitted. For this reason, consider taking a global snapshot so that information from all members is returned.

For example, Figure 23 shows the output of a **GET SNAPSHOT FOR APPLICATION** command, where the application holding the lock is on the same member as where the command is run:

```
ID of agent holding lock           = 73
Application ID holding lock        = *N0.user1.080616184956
Database partition lock wait occurred on = 0
Lock name                          = 0x0200040000000000000000000054
Lock attributes                    = 0x00000000
Release flags                      = 0x00000000
Lock object type                   = Table
Lock mode                          = Exclusive Lock (X)
Lock mode requested                = Share Lock (S)
Name of tablespace holding lock    = USERSPACE1
Schema of table holding lock       = USER1
Name of table holding lock         = T1
Data Partition Id of table holding lock = 0
Lock wait start timestamp          = 06/16/2009 14:50:26.744694
```

*Figure 23. Output for GET SNAPSHOT FOR APPLICATION command - command run on member holding lock.* In this example, the application holding the lock is running on the same member on which the **GET SNAPSHOT** command is being run:

However, if the lock is held by an application on a remote member, the same report would look like what is shown in Figure 24 on page 1482:

```

Application ID holding lock      = REMOTE APPLICATION
Database partition lock wait occurred on = 0
Lock name                       = 0x0200040000000000000000000054
Lock attributes                 = 0x00000000
Release flags                   = 0x00000000
Lock object type                = Table
Lock mode requested             = Share Lock (S)
Name of tablespace holding lock = USERSPACE1
Schema of table holding lock    = USER1
Name of table holding lock      = T1
Data Partition Id of table holding lock = 0
Lock wait start timestamp      = 06/16/2009 14:50:26.744694

```

*Figure 24. Output for GET SNAPSHOT FOR APPLICATION command - command run on member other than one holding lock. In this example, the application holding the lock is running on a different member on which the GET SNAPSHOT command is being run. The ID of agent holding lock and the Lock mode lines are omitted. Also, the Application ID holding lock is shown as REMOTE APPLICATION.*

If you take a global snapshot, data for all members is returned. You can determine where the lock is being held by examining the lock name from the snapshot output. If you skim through the reports for each of the members, you can quickly find which application holds the lock in question.

## Deadlock event monitor

When a deadlock occurs, the deadlock detector generates information used by event monitors that track deadlocks. The **CREATE EVENT MONITOR ... FOR DEADLOCKS** command, which is deprecated, might not show certain details for deadlocks between members in a DB2 pureScale environment. For deadlock event reporting, the details regarding the application that was the victim of the deadlock might not be shown in the output of the **dbevmon** tool, as shown in Figure 25 and in Figure 26: (Whether application details are shown or not is dependent on several factors,

```

3) Deadlock Event ...
  Deadlock ID: 1
  Deadlock node: 0
  Number of applications deadlocked: 2
  Deadlock detection time: 06/17/2009 14:46:22.543136
  Rolled back Appl participant no: 2

```

*Figure 25. Sample db2evmon output, DB2 pureScale instances*

```

3) Deadlock Event ...
  Deadlock ID: 1
  Deadlock node: 0
  Number of applications deadlocked: 2
  Deadlock detection time: 06/17/2009 14:46:22.543136
  Rolled back Appl participant no: 2
  Rolled back Appl Id: *N0.finance.081217170042
  Rolled back Appl seq number: : 0001
  Rolled back Appl handle: 66

```

*Figure 26. Sample db2evmon output, all other types of DB2 instances*

none of which are under user control.) However, you can determine which applications are involved in the deadlock by correlating the data reported for the deadlock ID and the rolled-back application participant number with information found in the Deadlocked Connection section of the **dbevmon** output.

Similarly, the Deadlocked Connection section of the output does not include the application ID, sequence number, or lock mode of the application holding the requested lock. However, the deadlock ID, member ID, lock name, lock time

stamp, and participant number of the application holding the lock are displayed. (The member ID appears as “Deadlock node” in the output of the **dbevmon** tool.) You can use this information to correlate which applications are causing the contention. The following sample output from the **db2evmon** tool illustrates this behavior. The information to use for correlating the applications involved in the deadlock is underlined:

```

5) Deadlock Event ...
  Deadlock ID: 1
  Deadlock node: 0
  Number of applications deadlocked: 2
  Deadlock detection time: 12/17/2008 12:01:12.735436
  Rolled back Appl participant no: 2
  Rolled back Appl Id: *N0.finance.081217170042
  Rolled back Appl seq number: : 0001
  Rolled back Appl handle: 66

6) Connection Header Event ...
  Appl Handle: 66
  Appl Id: *N0.finance.081217170042
  Appl Seq number: 00001
  DRDA AS Correlation Token: *N0.finance.081217170042
  Program Name : db2bp
  Authorization Id: FINANCE
  Execution Id : finance
  Codepage Id: 1208
  Territory code: 1
  Client Process Id: 7201
  Client Database Alias: A
  Client Product Id: SQL09070
  Client Platform: Unknown
  Client Communication Protocol: Local
  Client Network Name: so2.torolab.ibm.com
  Connect timestamp: 12/17/2008 12:00:42.176747

7) Deadlocked Connection ...
  Deadlock ID: 1
  Deadlock Node: 0
  Participant no.: 2
  Participant no. holding the lock: 1
  Appl Id: *N0.finance.081217170042
  Appl Seq number: 00001
  Appl Id of connection holding the lock: REMOTE APPLICATION
  Lock wait start time: 12/17/2008 12:01:01.607230
  Lock Name : 0x020005000400000100000000052
  Lock Attributes : 0x00000000
  Release Flags : 0x00000000
  Lock Count : 0
  Hold Count : 0
  Current Mode : none
  Deadlock detection time: 12/17/2008 12:01:17.730069
  Table of lock waited on : T2
  Schema of lock waited on : FINANCE
  Data partition id for table : 0
  Tablespace of lock waited on : USERSPACE1
  Type of lock: Row
  Mode application requested on lock: NS - Share (CS/RS)
  Node lock occurred on: 2
  Lock object name: 16777220
  Application Handle: 66
  Deadlocked Statement:
    Type : Dynamic
    Operation: Fetch
    Section : 201
    Creator : NULLID
    Package : SQLC2G17
    Cursor : SQLCUR201
    Cursor was blocking: FALSE
    Text : select * from t2

```

List of Locks:

...

```

Database partition : 0
Lock Name : 0x020004000100FFFFFFF81000000000052
Lock Attributes : 0x00000008
Release Flags : 0x40000000
Lock Count : 1

```

```

Hold Count          : 0
Lock Object Name    : 8454145
Object Type         : Row
Tablespace Name     : USERSPACE1
Table Schema        : FINANCE
Table Name          : T1
Data partition id   : 0
Mode                : X - Exclusive

Database partition  : 0
Lock Name           : 0x020005000000000000000000000000054
Lock Attributes     : 0x00000000
Release Flags       : 0x00000001
Lock Count          : 1
Hold Count          : 0
Lock Object Name    : 5
Object Type         : Table
Tablespace Name     : USERSPACE1
Table Schema        : FINANCE
Table Name          : T2
Data partition id   : 0
Mode                : IS - Intent Share
...

Locks Held: 6
Locks in List: 6
Locks Displayed: 6

```

```

8) Connection Header Event ...
AppI Handle: 131137
AppI Id: *N2.finance.081217170053
AppI Seq number: 00001
DRDA AS Correlation Token: *N2.finance.081217170053
Program Name : db2bp
Authorization Id: finance
Execution Id : finance
Codepage Id: 1208
Territory code: 1
Client Process Id: 7260
Client Database Alias: A
Client Product Id: SQL09070
Client Platform: Unknown
Client Communication Protocol: Local
Client Network Name: so2.torolab.ibm.com
Connect timestamp: 12/17/2008 12:00:43.542242

```

```

9) Deadlocked Connection ...
Deadlock ID: 1
Deadlock Node: 0
Participant no.: 1
Participant no. holding the lock: 2
AppI Id: *N2.finance.081217170053
AppI Seq number: 00001
AppI Id of connection holding the lock: REMOTE_APPLICATION
Lock wait start time: 12/17/2008 12:00:57.844388
Lock Name : 0x020004000100FFFFFFF810000000000052
Lock Attributes : 0x00000000
Release Flags : 0x00000000
Lock Count : 0
Hold Count : 0
Current Mode : none
Deadlock detection time: 12/17/2008 12:01:17.744611
Table of lock waited on : T1
Schema of lock waited on : FINANCE
Data partition id for table : 0
Tablespace of lock waited on : USERSPACE1
Type of lock: Row
Mode application requested on lock: NS - Share (CS/RS)
Node lock occurred on: 0
Lock object name: 8454145
Application Handle: 131137
Deadlocked Statement:
Type : Dynamic
Operation: Fetch
Section : 201
Creator : NULLID
Package : SQLC2G17
Cursor : SQLCUR201
Cursor was blocking: FALSE
Text : select * from t1

```

List of Locks:

...

```
Database partition      : 2
Lock Name               : 0x02000500040000010000000052
Lock Attributes         : 0x00000008
Release Flags          : 0x40000000
Lock Count              : 1
Hold Count              : 0
Lock Object Name       : 16777220
Object Type             : Row
Tablespace Name        : USERSPACE1
Table Schema           : FINANCE
Table Name              : T2
Data partition id      : 0
Mode                   : X - Exclusive
```

```
Database partition      : 2
Lock Name               : 0x02000500000000000000000054
Lock Attributes         : 0x00000000
Release Flags          : 0x40000000
Lock Count              : 1
Hold Count              : 0
Lock Object Name       : 5
Object Type             : Table
Tablespace Name        : USERSPACE1
Table Schema           : FINANCE
Table Name              : T2
Data partition id      : 0
Mode                   : IX - Intent Exclusive
```

```
Database partition      : 2
Lock Name               : 0x02000400000000000000000054
Lock Attributes         : 0x00000000
Release Flags          : 0x00000001
Lock Count              : 1
Hold Count              : 0
Lock Object Name       : 4
Object Type             : Table
Tablespace Name        : USERSPACE1
Table Schema           : FINANCE
Table Name              : T1
Data partition id      : 0
Mode                   : IS - Intent Share
```

Locks Held: 6  
Locks in List: 6  
Locks Displayed: 6

## LIST TABLESPACES and LIST TABLESPACE CONTAINERS commands

Both the **LIST TABLESPACES** and **LIST TABLESPACE CONTAINERS** commands were deprecated in DB2 Version 9.7. These commands report only information known on the member they are run on. They do not retrieve information from other members in the instance. So, some data, such as the number of used pages in a table space might not be reported accurately. Use the **MON\_GET\_TABLESPACE** and **MON\_GET\_CONTAINER** table functions instead.



---

## Chapter 15. New and changed monitor elements

---

### cf\_wait\_time - cluster caching facility wait time monitor element

The amount of time spent communicating with the cluster caching facility. This time does not include time spent performing any of the processing that may have been requested by, or that may occur as a result of the communications, such as granting locks or performing page reclaims. The unit of measurement is milliseconds.

*Table 2130. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                     | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times                   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache    | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                             | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                     | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |

#### Usage

This value is an indicator of the amount of time DB2 has spent waiting while communicating with the cluster caching facility.

---

### cf\_waits - Number of cluster caching facility waits monitor element

The number of times that the DB2 database system waited while it communicated with a cluster caching facility.

*Table 2131. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_CONNECTION table function - Get connection metrics                     | REQUEST METRICS BASE   |

Table 2131. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics                          | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics                                  | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE             |

Table 2132. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## configured\_cf\_gbp\_size - Configured cluster caching facility group buffer pool size monitor element

The allocated and reserved group buffer pool memory specified using the **cf\_gbp\_sz** configuration parameter, in pages with a page size of 4 KB.

Table 2133. Table Function Monitoring Information

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

## configured\_cf\_lock\_size - Configured cluster caching facility lock size monitor element

Global lock memory configured, in pages with a page size of 4 KB. This value is specified using the **cf\_lock\_sz** configuration parameter.

Table 2134. Table Function Monitoring Information

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## configured\_cf\_mem\_size - Configured cluster caching facility memory size monitor element

Configured total memory size for the cluster caching facility, in pages with a page size of 4 KB. This value is specified using the **cf\_mem\_sz** configuration parameter.

*Table 2135. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

---

## configured\_cf\_sca\_size - Configured cluster caching facility shared communications area size monitor element

Shared communications area memory currently allocated and reserved, in pages with a page size of 4 KB. This value is specified using the **cf\_sca\_sz** configuration parameter.

*Table 2136. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

---

## current\_cf\_gbp\_size - Current cluster caching facility group buffer pool size monitor element

Group buffer pool memory currently in use at the cluster caching facility, in pages with a page size of 4 KB.

*Table 2137. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

---

## current\_cf\_lock\_size - Current cluster caching facility lock size monitor element

Global lock memory currently in use, in pages with a page size of 4 KB.

*Table 2138. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

---

## current\_cf\_mem\_size - Current cluster caching facility memory size monitor element

Total memory currently in use, in pages with a page size of 4 KB.

*Table 2139. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## current\_cf\_sca\_size - Current cluster caching facility shared communications area size monitor element

Shared communications area memory currently in use, in pages with a page size of 4 KB.

*Table 2140. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## db\_name - Database name monitor element

The real name of the database for which information is collected or to which the application is connected. This is the name the database was given when created.

*Table 2141. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics  | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information                        | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function - Get memory set information                          | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics                     | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample workload metrics                               | ACTIVITY METRICS BASE            |

*Table 2142. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |
| Application    | appl_id_info          | Basic          |
| Application    | appl_remote           | Basic          |
| Table Space    | tablespace_list       | Buffer Pool    |

Table 2142. Snapshot Monitoring Information (continued)

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Buffer Pool     | bufferpool            | Buffer Pool    |
| Table           | table_list            | Table          |
| Lock            | db_lock_list          | Basic          |
| Dynamic SQL     | dynsql_list           | Basic          |
| DCS Database    | dcs_dbase             | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 2143. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Database   | event_dbheader        | Always collected |

## Usage

You may use this element to identify the specific database to which the data applies.

For applications that are not using DB2 Connect to connect to a host or System i database server, you can use this element in conjunction with the **db\_path** monitor element to uniquely identify the database and help relate the different levels of information provided by the monitor.

---

## dbpartitionnum - Database partition number monitor element

In a partitioned database environment, this is the numeric identifier for the database member. For DB2 Enterprise Server Edition and in a DB2 pureScale environment, this value is 0.

Table 2144. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO table function - returns compressed index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function - returns index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_MSGS table function - Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure | ACTIVITY METRICS BASE            |
| ADMIN_GET_STORAGE_PATHS table function - Get storage path information for storage groups  | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries   | ACTIVITY METRICS BASE            |

Table 2144. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information    | ACTIVITY METRICS BASE            |
| AUDIT_ARCHIVE procedure and table function - Archive audit log file   | ACTIVITY METRICS BASE            |
| DBCFC administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information       | ACTIVITY METRICS BASE            |
| DBPATHS administrative view and ADMIN_LIST_DB_PATHS table function - Retrieve database paths                          | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function - Get table space container metrics  | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS table function - Get rebalance progress for a table space                                    | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics   | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility  | ACTIVITY METRICS BASE            |
| PDLOGMSG_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function - Retrieve problem determination messages | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                                  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                                 | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses                               | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUPERCLASS_STATS table function - Return statistics of service superclasses                           | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS table function - Return work action set statistics                                      | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function - Return a list of activities                                   | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table function - Return workload statistics  | ACTIVITY METRICS BASE            |

## Usage

In a DB2 pureScale environment, multiple members operate on a single partition. When running in such a configuration, physical attributes of storage, such as the number of free pages in a table space, are duplicated across all members in the system. Each member reports the total accurate size for the system. In a multiple partition configuration, the values from each partition must be correlated by the user in order to understand the overall value for the system.

The **dbpartitionnum** monitor element is different to the **data\_partition\_id** monitor element, which is used to identify a data partition that was created by subdividing data in a table based on a value.

---

## host\_name - Host name monitor element

Name of the host on which the cluster caching facility process resides.

*Table 2145. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| DB_MEMBERS table function   | ACTIVITY METRICS BASE            |
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information | ACTIVITY METRICS BASE            |
| ENV_GET_SYSTEM_RESOURCES table function - Return system information           | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics                                    | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function - Get memory pool information              | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function - Get memory set information                | ACTIVITY METRICS BASE            |

---

## id - cluster caching facility identification monitor element

cluster caching facility identifier, as defined in the `db2nodes.cfg` file.

*Table 2146. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| DB2_MEMBER and DB2_CF administrative views and DB2_GET_INSTANCE_INFO table function  | ACTIVITY METRICS BASE            |
| MON_GET_CF table function - Get CF metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CF_CMD table function - Get cluster caching facility command processing time | ACTIVITY METRICS BASE            |
| MON_GET_CF_WAIT_TIME table function - Get cluster caching facility command wait time | ACTIVITY METRICS BASE            |

## lock\_escals - Number of lock escalations monitor element

The number of times that locks have been escalated from several row locks to a table lock.

*Table 2147. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries                             | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2148. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 2149. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Database      | event_db  | Always collected      |
| Connection    | event_conn  | Always collected      |
| Transactions  | event_xact  | Always collected      |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the **maxlocks** and **locklist** configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs.

This data item includes a count of all lock escalations, including exclusive lock escalations and escalations in the DB2 pureScale environment. To determine just the lock escalations in the DB2 pureScale environment, use the **lock\_escals\_global** monitor element.

There are several possible causes for excessive lock escalations:

- The lock list size (**locklist**) may be too small for the number of concurrent applications
- The percent of the lock list usable by each application (**maxlocks**) may be too small
- One or more applications may be using an excessive number of locks.
- In the DB2 pureScale environment, the global lock list size (**cf\_lock\_sz**) may be too small.

To resolve these problems, you may be able to:

- Increase the **locklist** configuration parameter value.
- Increase the **maxlocks** configuration parameter value.
- Identify the applications with large numbers of locks, or those that are holding too much of the lock list, using one of the following formulae, and comparing the value to **maxlocks**.

- On 64-bit systems,  $((locks\ held * 64) / (locklist * 4096)) * 100$
- On 32-bit systems,  $((locks\ held * 48) / (locklist * 4096)) * 100$

These applications can also cause lock escalations in other applications by using too large a portion of the lock list. These applications may need to resort to using table locks instead of row locks, although table locks may cause an increase in **lock\_waits** and **lock\_wait\_time** monitor element values.

---

## lock\_escals\_global - Number of global lock escalations monitor element

Number of lock escalations on a global lock due to global lock memory usage reaching the limit specified in the **cf\_lock\_sz** database configuration parameter.

*Table 2150. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 2151. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element together with the **lock\_escals\_maxlocks** and **lock\_escals\_locklist** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

## lock\_escals\_locklist - Number of locklist lock escalations monitor element

Number of lock escalations due to local lock memory usage reaching the limit specified in the **locklist** database configuration parameter.

Table 2152. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 2152. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 2153. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element together with the **lock\_escals\_maxlocks** and **lock\_escals\_global** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

## lock\_escals\_maxlocks - Number of maxlocks lock escalations monitor element

Number of lock escalations due to local lock memory usage reaching the limit specified in the **maxlocks** database configuration parameter.

Table 2154. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 2154. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 2155. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element together with the **lock\_escals\_locklist** and **lock\_escals\_global** monitor elements to determine which lock space configuration parameter is causing escalations on the database.

## lock\_timeouts\_global - Lock timeouts global monitor element

Number of lock timeouts where the application holding the lock was on a remote member.

*Table 2156. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2157. Event Monitoring Information*

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this element in conjunction with the **lock\_timeouts** monitor element. The **lock\_timeouts\_global** monitor element represents the number of times a lock timeout has occurred while waiting to acquire a lock held on another member. To determine the number of times a lock timeout has occurred while waiting to acquire a lock held on the same member, use the following formula:

`lock_timeouts - lock_timeouts_global`

Outside of the DB2 pureScale environment, this value is always zero.

---

## lock\_wait\_time\_global - Lock wait time global monitor element

Time spent on global lock waits. The unit of measurement for time is in milliseconds.

*Table 2158. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |

Table 2158. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |

Table 2159. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

Use this monitor element in conjunction with the **lock\_wait\_time** monitor element, which represents all the time spent waiting for locks. The **lock\_wait\_time\_global** monitor element represents the time spent waiting for locks held by conflicting applications on different members. To determine the total time spent waiting for locks held by conflicting applications on the same member, use the following formula:

$$\text{lock\_wait\_time} - \text{lock\_wait\_time\_global}$$

Outside of the DB2 pureScale environment, this value is always zero.

## lock\_wait\_time\_global\_top - Top global lock wait time monitor element

The longest lock wait that has occurred for a lock that is held on another member. This value is reported in milliseconds.

Table 2160. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch   |
|------------|-----------------------|------------------|
| Statistics | event_wlstats         | Always collected |

## lock\_waits\_global - Lock waits global monitor element

Number of lock waits due to the application holding the lock being on a remote member.

*Table 2161. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLE table function - get table metrics   | DATA OBJECT METRICS EXTENDED   |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                      | DATA OBJECT METRICS EXTENDED   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2162. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2162. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

Use this monitor element in conjunction with the **lock\_waits** monitor element, which reports the total number of lock waits due to locks held by conflicting applications on all members. The **lock\_waits\_global** monitor element indicates the number of times that a lock wait was held by conflicting applications on different members. To determine the number of lock waits held by a conflicting application on the same member as the waiting application, use the following formula:

`lock_waits - lock_waits_global`

Outside of the DB2 pureScale environment, this value is always zero.

---

## member - Database member monitor element

The numeric identifier for the database member from which the data was retrieved for this result record.

Table 2163. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_MEM_USAGE table function - Get total memory consumption for instance                                  | ACTIVITY METRICS BASE            |
| AUDIT_ARCHIVE procedure and table function - Archive audit log file   | ACTIVITY METRICS BASE            |
| DBCFC administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information | ACTIVITY METRICS BASE            |
| ENV_GET_REG_VARIABLES table function - Retrieve DB2 registry settings in use                                    | ACTIVITY METRICS BASE            |
| ENV_GET_DB2_SYSTEM_RESOURCES table function - Return DB2(r) system information                                  | ACTIVITY METRICS BASE            |
| ENV_GET_SYSTEM_RESOURCES table function - Return system information   | ACTIVITY METRICS BASE            |
| ENV_GET_NETWORK_RESOURCES table function - Return network adapter information                                   | ACTIVITY METRICS BASE            |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details   | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs                         | ACTIVITY METRICS BASE            |

Table 2163. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_AUTO_RUNSTATS_QUEUE<br>table function - Retrieve information about<br>objects queued for evaluation | ACTIVITY METRICS BASE            |
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics  | ACTIVITY METRICS BASE            |
| MON_GET_CF_WAIT_TIME table function -<br>Get cluster caching facility command wait<br>time                  | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION table function -<br>Get connection metrics   | ACTIVITY METRICS BASE            |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics                              | ACTIVITY METRICS BASE            |
| MON_GET_CONTAINER table function -<br>Get table space container metrics                                     | ACTIVITY METRICS BASE            |
| MON_GET_EXTENDED_LATCH_WAIT table<br>function - Return information for latches                              | ACTIVITY METRICS BASE            |
| MON_GET_EXTENT_MOVEMENT_STATUS<br>- Get extent movement progress status<br>metrics                          | ACTIVITY METRICS BASE            |
| MON_GET_FCM - Get FCM metrics   | ACTIVITY METRICS BASE            |
| MON_GET_FCM_CONNECTION_LIST - Get<br>details for all FCM connections  | ACTIVITY METRICS BASE            |
| MON_GET_GROUP_BUFFERPOOL table<br>function - Get group buffer pool metrics                                  | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index<br>metrics   | ACTIVITY METRICS BASE            |
| MON_GET_INDEX_USAGE_LIST table<br>function - Returns information from an index<br>usage list                | ACTIVITY METRICS BASE            |
| MON_GET_LOCKS table function - list all<br>locks in the currently connected database                        | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_POOL table function -<br>Get memory pool information   | ACTIVITY METRICS BASE            |
| MON_GET_MEMORY_SET table function -<br>Get memory set information   | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table<br>function - Get buffer pool page waiting<br>information                    | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache       | ACTIVITY METRICS BASE            |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries        | ACTIVITY METRICS BASE            |
| MON_GET_REBALANCE_STATUS table<br>function - get rebalance progress for a table<br>space                    | ACTIVITY METRICS BASE            |

Table 2163. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests                            | ACTIVITY METRICS BASE            |
| MON_GET_SERVERLIST table function - get member priority details   | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | ACTIVITY METRICS BASE            |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics                               | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list                                 | ACTIVITY METRICS BASE            |
| MON_GET_TABLESPACE table function - Get table space metrics   | ACTIVITY METRICS BASE            |
| MON_GET_TRANSACTION_LOG table function - Get log information  | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | ACTIVITY METRICS BASE            |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics                                       | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list   | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD table function - Get workload metrics  | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics   | ACTIVITY METRICS BASE            |
| MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics   | ACTIVITY METRICS BASE            |
| MON_SAMPLE_WORKLOAD_METRICS - Get sample  | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility  | ACTIVITY METRICS BASE            |
| PDLOGMSG_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function - Retrieve problem determination messages | ACTIVITY METRICS BASE            |
| WLM_GET_QUEUE_STATS table function - Return threshold queue statistics  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class                                  | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function - list workload occurrences                                 | ACTIVITY METRICS BASE            |

Table 2163. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| WLM_GET_SERVICE_SUBCLASS_STATS<br>table function - Return statistics of service subclasses     | ACTIVITY METRICS BASE            |
| WLM_GET_SERVICE_SUPERCLASS_STATS<br>table function - Return statistics of service superclasses | ACTIVITY METRICS BASE            |
| WLM_GET_WORK_ACTION_SET_STATS<br>table function - Return work action set statistics            | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_OCCURRENCE<br>_ACTIVITIES table function - Return a list of activities        | ACTIVITY METRICS BASE            |
| WLM_GET_WORKLOAD_STATS table<br>function - Return workload statistics                          | ACTIVITY METRICS BASE            |

Table 2164. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Switch        |
|----------------|--|-----------------------|
| Activities     | event_activity (reported in the details_xml document)  | ACTIVITY METRICS BASE |
| Statistics     | event_scstats (reported in the details_xml document)   | REQUEST METRICS BASE  |
| Statistics     | event_wlstats (reported in the details_xml document)   | REQUEST METRICS BASE  |
| Unit of work   | Reported in the system_metrics document.   | REQUEST METRICS BASE  |
| Package cache  | -  | COLLECT BASE DATA     |
| Locking        | -  | Always collected      |
| Change history | changesummary<br>dbdbmcfg<br>regvar<br>ddlstmtexec<br>txncompletion<br>evmonstart<br>utilstart<br>utillocation<br>utilstop | Always collected      |

## Usage

A DB2 member is a database manager instance that runs DB2 server software on a single host. A DB2 member accepts and processes database requests from applications connected to it.

## objtype - Object type monitor element

The type of object for which monitoring data is being reported. This monitor element is an alias of the object\_type monitor element.

Table 2165. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings                             | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries       | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_MAINT_QUEUE table function - Get information on automatic maintenance jobs               | ACTIVITY METRICS BASE            |
| MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information                    | ACTIVITY METRICS BASE            |
| MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests            | ACTIVITY METRICS BASE            |
| MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list                         | ACTIVITY METRICS BASE            |
| PD_GET_DIAG_HIST table function - Return records from a given facility                                | ACTIVITY METRICS BASE            |

Table 2166. Event Monitoring Information

| Event Type     | Logical Data Grouping  | Monitor Switch   |
|----------------|------------------------|------------------|
| Change history | utilphase<br>utilstart | Always collected |

### Usage Notes

- When this element is returned by the ADMIN\_GET\_TAB\_COMPRESS\_INFO table function or the ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function, the returned value for the object\_type monitor element can be either 'XML' or 'DATA'.
- When this element is returned by the MON\_GET\_RTS\_RQST table function, the returned value for the object\_type monitor element is 'TABLE'.
- When this element is returned by the MON\_GET\_AUTO\_MAINT\_QUEUE table function, the returned value for the object\_type monitor element can be 'DATABASE', 'TABLE', 'NICKNAME', or 'VIEW'.
- When this element is returned by the MON\_GET\_AUTO\_RUNSTATS\_QUEUE table function, the returned value for the object\_type monitor element can be 'TABLE', 'NICKNAME', or 'VIEW'.
- When this element is returned by the change history event monitor, the returned value for the object\_type monitor element can be 'DATABASE', 'INDEX', 'PARTITIONGROUP', 'TABLE' or 'TABLESPACE'.

---

## page\_reclaims\_initiated\_s - Page reclaims initiated shared access monitor element

The number of times a page accessed in shared mode caused the page to be reclaimed from another member. Page reclaims related to internally maintained object space maps are counted separately.

*Table 2167. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

---

## page\_reclaims\_initiated\_x - Page reclaims initiated exclusive access monitor element

The number of times a page accessed in exclusive mode caused the page to be reclaimed from another member. Page reclaims related to internally maintained object space maps are counted separately.

*Table 2168. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

---

## page\_reclaims\_s - Page reclaims shared access monitor element

The number of times a page related to the object was reclaimed by another member in the DB2 pureScale instance before its planned release, where the member that reclaimed the page required shared access.

*Table 2169. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

### Usage

Page reclaims related to internally maintained object space maps are counted separately.

---

## page\_reclaims\_x - Page reclaims exclusive access monitor element

The number of times a page related to the object was reclaimed by another member in the DB2 pureScale instance before its planned release, where the member that reclaimed the page required exclusive access.

*Table 2170. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

### Usage

Page reclaims related to internally maintained object space maps are counted separately.

---

## pool\_async\_data\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid data pages monitor element

The number of times a data page was attempted to be read from the group buffer pool by a prefetcher because the page was invalid in the local buffer pool.

*Table 2171. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function - Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \frac{(\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads})}{\text{pool\_async\_data\_gbp\_l\_reads}}$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_data\_gbp\_l\_reads - Asynchronous group buffer pool data logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 2172. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics | DATA OBJECT METRICS BASE         |

Table 2172. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}}{\text{pool\_async\_data\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_data\_gbp\_p\_reads - Asynchronous group buffer pool data physical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

Table 2173. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}}{\text{pool\_async\_data\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_data\_lbp\_pages\_found - Asynchronous local buffer pool data pages found monitor element

The number of times a data page was present in the local buffer pool when a prefetcher attempted to access it.

Table 2174. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) data page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_data\_gbp\_l\_reads} - \text{pool\_async\_data\_gbp\_p\_reads}}{\text{pool\_async\_data\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the IBM DB2 pureScale Feature. Using this formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid index pages monitor element

The number of times an index page was attempted to be read from the group buffer pool by a prefetcher because the page was invalid in the local buffer pool.

*Table 2175. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads}}{\text{pool\_async\_index\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_gbp\_l\_reads - Asynchronous group buffer pool index logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 2176. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

## Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = ( \text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads} ) / \text{pool\_async\_index\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_gbp\_p\_reads - Asynchronous group buffer pool index physical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

*Table 2177. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = ( \text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads} ) / \text{pool\_async\_index\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_index\_lbp\_pages\_found - Asynchronous local buffer pool index pages found monitor element

The number of times an index page was present in the local buffer pool when a prefetcher attempted to access it.

*Table 2178. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) index page hit ratios can be computed as follows:

$$\text{GBP} = ( \text{pool\_async\_index\_gbp\_l\_reads} - \text{pool\_async\_index\_gbp\_p\_reads} ) / \text{pool\_async\_index\_gbp\_l\_reads}$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_gbp\_invalid\_pages - Asynchronous group buffer pool invalid XDA data pages monitor element

The number of times a request for a data page for an XML storage object (XDA) was made from the group buffer pool by a prefetcher due to the page being marked invalid in the local buffer pool.

*Table 2179. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_gbp\_l\_reads - Group buffer pool XDA data asynchronous logical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was attempted to be read from the group buffer pool by a prefetcher because the page was either invalid or not present in the local buffer pool.

*Table 2180. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you

determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_gbp\_p\_reads - Group buffer pool XDA data asynchronous physical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was read into the local buffer pool by a prefetcher from disk because it was not found in the GBP.

*Table 2181. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_async\_xda\_lbp\_pages\_found - Asynchronous local buffer pool XDA data pages found monitor element

The number of times a data page for an XML storage object (XDA) was requested by a prefetcher from and found in the local buffer pool.

*Table 2182. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics | DATA OBJECT METRICS BASE         |
| MON_GET_TABLESPACE table function -<br>Get table space metrics | DATA OBJECT METRICS BASE         |

### Usage

The prefetcher (or asynchronous) XDA page hit ratios can be computed as follows:

$$\text{GBP} = \left( \frac{\text{pool\_async\_xda\_gbp\_l\_reads} - \text{pool\_async\_xda\_gbp\_p\_reads}}{\text{pool\_async\_xda\_gbp\_l\_reads}} \right)$$

Buffer pool hit rates are important factors in the overall performance of the DB2 pureScale instance. Using the previously mentioned formula can help you determine whether the group buffer pool may be a limiting factor on your database's throughput.

## pool\_data\_gbp\_invalid\_pages - Group buffer pool invalid data pages monitor element

The number of times a data page was invalid in the local buffer pool and was read from the group buffer pool instead. Outside of a DB2 pureScale environment, this value is null.

Table 2183. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 2184. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |

Table 2184. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_scstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_data\_gbp\_l\_reads - Group buffer pool data logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was attempted to be read from the group buffer pool because the page was either invalid or not present in the Local Buffer Pool (LBP).

Table 2185. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |

Table 2185. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 2186. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_data\_gbp\_p\_reads - Group buffer pool data physical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent data page was read into the local buffer pool from disk because it was not found in the GBP.

Table 2187. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 2188. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2188. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_data\_lbp\_pages\_found - Local buffer pool found data pages monitor element

The number of times a data page was present in the local buffer pool.

Table 2189. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 2189. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 2190. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -  | Always collected      |
| Package cache | -  | Always collected      |
| Locking       | -  | Always collected      |

## Usage

To determine how often a requested page was found in the local buffer pool, use the following formula:

$$(POOL\_DATA\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_DATA\_LBP\_PAGES\_FOUND) / POOL\_DATA\_L\_READS$$

To determine how many times a requested page was found in the group bufferpool, use the following formula

$$(POOL\_DATA\_GBP\_L\_READS - POOL\_DATA\_GBP\_P\_READS) / POOL\_DATA\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_index\_gbp\_invalid\_pages - Group buffer pool invalid index pages monitor element

The number of times an index page was attempted to be read from the group bufferpool because the page was invalid in the local buffer pool.

*Table 2191. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2192. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2192. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_index\_gbp\_l\_reads - Group buffer pool index logical reads monitor element

The number of times a Group Buffer Pool (GBP) dependent index page was attempted to be read from the group buffer pool because the page was either invalid or not present in the local buffer pool.

Table 2193. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 2193. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE             |

Table 2194. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -   | Always collected      |
| Package cache | -   | Always collected      |
| Locking       | -   | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_index\_gbp\_p\_reads - Group buffer pool index physical reads monitor elements

The number of times a Group Buffer Pool (GBP) dependent index page was read into the local buffer pool from disk because it was not found in the GBP.

*Table 2195. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

*Table 2196. Event Monitoring Information*

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2196. Event Monitoring Information (continued)

| Event Type    | Logical Data Grouping                                | Monitor Switch       |
|---------------|--|----------------------|
| Statistics    | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work  | -  | Always collected     |
| Package cache | -  | Always collected     |
| Locking       | -  | Always collected     |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(POOL\_INDEX\_LBP\_PAGES\_FOUND - POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND) / POOL\_INDEX\_L\_READS$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(POOL\_INDEX\_GBP\_L\_READS - POOL\_INDEX\_GBP\_P\_READS) / POOL\_INDEX\_GBP\_L\_READS$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_index\_lbp\_pages\_found - Local buffer pool index pages found monitor element

The number of times an index page was present in the local buffer pool.

Table 2197. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |

Table 2197. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document) | REQUEST METRICS BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics  | DATA OBJECT METRICS BASE         |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)         | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics   | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                    | REQUEST METRICS BASE             |

Table 2198. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | -  | Always collected      |
| Package cache | -  | Always collected      |
| Locking       | -  | Always collected      |

## Usage

To determine how often a requested index page was found in the local buffer pool, use the following formula:

$$(\text{POOL\_INDEX\_LBP\_PAGES\_FOUND} - \text{POOL\_ASYNC\_INDEX\_LBP\_PAGES\_FOUND}) / \text{POOL\_INDEX\_L\_READS}$$

To determine how many times a requested index page was found in the group bufferpool, use the following formula

$$(\text{POOL\_INDEX\_GBP\_L\_READS} - \text{POOL\_INDEX\_GBP\_P\_READS}) / \text{POOL\_INDEX\_GBP\_L\_READS}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_xda\_gbp\_invalid\_pages - Group buffer pool invalid XDA data pages monitor element

The number of times a request for a data page for an XML storage object (XDA) was made from the group buffer pool due to the page being marked invalid in the local buffer pool.

*Table 2199. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE   |

Table 2200. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_xda\_gbp\_l\_reads - Group buffer pool XDA data logical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was attempted to be read from the group buffer pool because the page was either invalid or not present in the local buffer pool.

Table 2201. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                                 | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)     | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics  | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 2201. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element          | Collection Level |
|---|--------------------------|------------------|
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE    |                  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE    |                  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE     |                  |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE     |                  |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE |                  |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE     |                  |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE     |                  |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE     |                  |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                        | REQUEST METRICS BASE     |                  |

Table 2202. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(pool\_xda\_gbp\_l\_reads - pool\_xda\_gbp\_p\_reads) / pool\_xda\_gbp\_l\_reads$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## pool\_xda\_gbp\_p\_reads - Group buffer pool XDA data physical read requests monitor element

The number of times a GBP dependent data page for an XML storage object (XDA) was read into the local buffer pool from disk because it was not found in the group buffer pool.

*Table 2203. Table Function Monitoring Information*

| Table Function  | Monitor Element Collection Level   |
|---|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics  | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                        | ACTIVITY METRICS BASE  |
| MON_GET_BUFFERPOOL table function - Get buffer pool metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_CONNECTION table function - Get connection metrics  | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)                    | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                                   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT_DETAILS table function - get detailed metrics for package cache entries (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics  | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document)        | REQUEST METRICS BASE   |
| MON_GET_TABLESPACE table function - Get table space metrics   | DATA OBJECT METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics  | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)                | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics  | REQUEST METRICS BASE   |

Table 2203. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |

Table 2204. Event Monitoring Information

| Event Type    | Logical Data Grouping                                 | Monitor Switch        |
|---------------|---|-----------------------|
| Activities    | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

## pool\_xda\_lbp\_pages\_found - Local buffer pool XDA data pages found monitor element

The number of times a data page for an XML storage object (XDA) was requested from and found in the local buffer pool.

Table 2205. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics                             | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document) | ACTIVITY METRICS BASE  |

Table 2205. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element     | Collection Level |
|--|---------------------|------------------|
| MON_GET_BUFFERPOOL table function -<br>Get buffer pool metrics   | DATA OBJECT METRICS | BASE             |
| MON_GET_CONNECTION table function -<br>Get connection metrics  | REQUEST METRICS     | BASE             |
| MON_GET_CONNECTION_DETAILS table<br>function - Get detailed connection metrics<br>(reported in DETAILS XML document)                       | REQUEST METRICS     | BASE             |
| MON_GET_PKG_CACHE_STMT table<br>function - Get SQL statement activity metrics<br>in the package cache                                      | ACTIVITY METRICS    | BASE             |
| MON_GET_PKG_CACHE_STMT_DETAILS<br>table function - get detailed metrics for<br>package cache entries (reported in DETAILS<br>XML document) | ACTIVITY METRICS    | BASE             |
| MON_GET_SERVICE_SUBCLASS table<br>function - Get service subclass metrics  | REQUEST METRICS     | BASE             |
| MON_GET_SERVICE_SUBCLASS_DETAILS<br>table function - Get detailed service subclass<br>metrics (reported in DETAILS XML<br>document)        | REQUEST METRICS     | BASE             |
| MON_GET_TABLESPACE table function -<br>Get table space metrics   | DATA OBJECT METRICS | BASE             |
| MON_GET_UNIT_OF_WORK table function<br>- Get unit of work metrics  | REQUEST METRICS     | BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS<br>table function - Get detailed unit of work<br>metrics (reported in DETAILS XML<br>document)                | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD table function -<br>Get workload metrics  | REQUEST METRICS     | BASE             |
| MON_GET_WORKLOAD_DETAILS table<br>function - Get detailed workload metrics<br>(reported in DETAILS XML document)                           | REQUEST METRICS     | BASE             |

Table 2206. Event Monitoring Information

| Event Type    | Logical Data Grouping                                    | Monitor Switch        |
|---------------|--|-----------------------|
| Activities    | event_activity (reported in<br>the details_xml document) | ACTIVITY METRICS BASE |
| Activities    | event_activitymetrics                                    | ACTIVITY METRICS BASE |
| Statistics    | event_scstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Statistics    | event_wlstats (reported in the<br>details_xml document)  | REQUEST METRICS BASE  |
| Unit of work  | Reported in the<br>system_metrics document.              | REQUEST METRICS BASE  |
| Package cache | Reported in the<br>activity_metrics document.            | ACTIVITY METRICS BASE |

## Usage

To determine how often a requested XDA page was found in the local buffer pool, use the following formula:

$$(\text{pool\_xda\_lbp\_pages\_found} - \text{pool\_async\_xda\_lbp\_pages\_found}) / \text{pool\_xda\_l\_reads}$$

To determine how many times a requested XDA page was found in the group buffer pool, use the following formula

$$(\text{pool\_xda\_gbp\_l\_reads} - \text{pool\_xda\_gbp\_p\_reads}) / \text{pool\_xda\_gbp\_l\_reads}$$

Local buffer pool and group buffer pool hit rates are both important factors in the overall performance of the cluster caching facility. Using these formulas can help you determine whether the local or group buffer pool may be a limiting factor on your database's throughput.

---

## reclaim\_wait\_time - Reclaim wait time monitor element

In a DB2 pureScale environment, this element represents the amount of time spent waiting on page locks, where the lock request caused a page to be reclaimed. The unit of measurement for time is in milliseconds.

*Table 2207. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information   | ACTIVITY METRICS BASE  |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |

Table 2207. Table Function Monitoring Information (continued)

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE             |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document) | REQUEST METRICS BASE             |
| MON_GET_WORKLOAD table function - Get workload metrics   | ACTIVITY METRICS BASE            |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)         | ACTIVITY METRICS BASE            |

Table 2208. Event Monitoring Information

| Event Type   | Logical Data Grouping                                 | Monitor Switch        |
|--------------|---|-----------------------|
| Activities   | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities   | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics   | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Statistics   | event_wlstats (reported in the details_xml document)  | REQUEST METRICS BASE  |
| Unit of work | -   | Always collected      |

## Usage

The time spent waiting for reclaims on space map pages is counted separately and reported in the **spacemappage\_reclaim\_wait\_time** monitor element.

## spacemappage\_page\_reclaims\_initiated\_s - Space map page reclaims initiated shared access monitor element

The number of times a page accessed in shared mode of a space map page caused the page to be reclaimed from another member.

Table 2209. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_initiated\_s** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_initiated\_s** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_initiated\_s** monitor element.

---

## spacemappage\_page\_reclaims\_initiated\_x - Space map page reclaims initiated exclusive access monitor element

The number of times a page accessed in exclusive mode for a space map page caused the page to be reclaimed from another member.

*Table 2210. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

### Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_initiated\_x** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_initiated\_x** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_initiated\_x** monitor element.

---

## spacemappage\_page\_reclaims\_s - Space map page reclaims shared access monitor element

The number of times a page related to a space map page was reclaimed by another member in the DB2 pureScale instance before its planned release. The member that reclaimed the page required shared access to the space map page.

*Table 2211. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_s** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_s** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_s** monitor element.

---

## **spacemappage\_page\_reclaims\_x - Space map page reclaims exclusive access monitor element**

The number of times a page related to a space map page was reclaimed by another member in the DB2 pureScale instance before its planned release. The member that reclaimed the page required exclusive access to the space map page.

*Table 2212. Table Function Monitoring Information*

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information | ACTIVITY METRICS BASE            |

## Usage

This value is only reported for object-relative table spaces, that is table spaces that have been enabled for reclaimable storage. Use the **reclaimable\_space\_enabled** monitor element to determine if the table space has been enabled for reclaimable storage.

Since Extent Map Pages (EMPs) are metadata, EMPs are included in the value of this monitor element.

Data space map pages contain user data, therefore they are included in the value of the **page\_reclaims\_x** monitor element, in addition to being included the value of the **spacemappage\_page\_reclaims\_x** monitor element. Index space map pages do not contain user data, therefore they are included only in the value of the **spacemappage\_page\_reclaims\_x** monitor element.

---

## **spacemappage\_reclaim\_wait\_time - Space map page reclaim wait time monitor element**

In a DB2 pureScale environment, this element represents the amount of time spent waiting on page locks for pages related to internally maintained object space management where the lock request caused a reclaim from another member. The unit of measurement for time is in milliseconds.

Table 2213. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level   |
|--|--|
| MON_FORMAT_XML_METRICS_BY_ROW - Get formatted row-based output for all metrics   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_TIMES_BY_ROW - Get formatted row-based combined hierarchy wait and processing times                         | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_FORMAT_XML_WAIT_TIMES_BY_ROW - Get formatted row-based output for wait times   | Not applicable; reports whichever elements are contained in the XML document provided as input to the formatting function. |
| MON_GET_ACTIVITY_DETAILS table function - Get complete activity details (reported in DETAILS XML document)                 | ACTIVITY METRICS BASE  |
| MON_GET_CONNECTION table function - Get connection metrics   | REQUEST METRICS BASE   |
| MON_GET_CONNECTION_DETAILS table function - Get detailed connection metrics (reported in DETAILS XML document)             | REQUEST METRICS BASE   |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information   | REQUEST METRICS BASE   |
| MON_GET_PKG_CACHE_STMT table function - Get SQL statement activity metrics in the package cache                            | ACTIVITY METRICS BASE  |
| MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics   | REQUEST METRICS BASE   |
| MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get detailed service subclass metrics (reported in DETAILS XML document) | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK table function - Get unit of work metrics   | REQUEST METRICS BASE   |
| MON_GET_UNIT_OF_WORK_DETAILS table function - Get detailed unit of work metrics (reported in DETAILS XML document)         | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD table function - Get workload metrics   | REQUEST METRICS BASE   |
| MON_GET_WORKLOAD_DETAILS table function - Get detailed workload metrics (reported in DETAILS XML document)                 | REQUEST METRICS BASE   |

Table 2214. Event Monitoring Information

| Event Type | Logical Data Grouping                                 | Monitor Switch        |
|------------|---|-----------------------|
| Activities | event_activity (reported in the details_xml document) | ACTIVITY METRICS BASE |
| Activities | event_activitymetrics                                 | ACTIVITY METRICS BASE |
| Statistics | event_scstats (reported in the details_xml document)  | REQUEST METRICS BASE  |

Table 2214. Event Monitoring Information (continued)

| Event Type   | Logical Data Grouping                                | Monitor Switch       |
|--------------|--|----------------------|
| Statistics   | event_wlstats (reported in the details_xml document) | REQUEST METRICS BASE |
| Unit of work | -  | Always collected     |

## table\_name - Table name monitor element

The name of the table.

Table 2215. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO table function - returns compressed index information  | ACTIVITY METRICS BASE            |
| ADMIN_GET_INDEX_INFO table function - returns index information  | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings  | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries                                    | ACTIVITY METRICS BASE            |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information                 | ACTIVITY METRICS BASE            |
| ADMINTEMP_COLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS table function - Retrieve column information for temporary tables | ACTIVITY METRICS BASE            |
| ADMINTEMP_TABLES administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables          | ACTIVITY METRICS BASE            |
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details   | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics   | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information   | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics   | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list  | ACTIVITY METRICS BASE            |

Table 2216. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 2216. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 2217. Event Monitoring Information

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Locking                             | -                     | Always collected |
| Tables                              | event_table           | Always collected |
| Deadlocks <sup>1</sup>              | lock                  | Always collected |
| Deadlocks <sup>1</sup>              | event_dlconn          | Always collected |
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

- 1** This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Along with **table\_schema**, this element can help you determine the source of contention for resources.

At the application-level, application-lock level, and deadlock-monitoring-level, this is the table that the application is waiting to lock, because it is currently locked by another application. For snapshot monitoring, this item is only valid when the “lock” monitor group information is set to ON, and when **lock\_object\_type** indicates that the application is waiting to obtain a table lock.

For snapshot monitoring at the object-lock level, this item is returned for table-level and row-level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this is the table for which information has been collected. For temporary tables, the format for **table\_name** is “TEMP (*n*, *m*)”, where:

- *n* is the table space ID
- *m* is the **table\_file\_id** element

---

## table\_schema - Table schema name monitor element

The schema of the table.

Table 2218. Table Function Monitoring Information

| Table Function   | Monitor Element Collection Level |
|--|----------------------------------|
| ADMIN_GET_INDEX_COMPRESS_INFO<br>table function - returns compressed index information | ACTIVITY METRICS BASE            |

Table 2218. Table Function Monitoring Information (continued)

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| ADMIN_GET_INDEX_INFO table function - returns index information   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings   | ACTIVITY METRICS BASE            |
| ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries                                   | ACTIVITY METRICS BASE            |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information                | ACTIVITY METRICS BASE            |
| ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS table function - Retrieve column information for temporary tables | ACTIVITY METRICS BASE            |
| ADMINTEMPTables administrative view and ADMIN_GET_TEMP_TABLES table function - Retrieve information for temporary tables          | ACTIVITY METRICS BASE            |
| MON_FORMAT_LOCK_NAME table function - Format the internal lock name and return details  | ACTIVITY METRICS BASE            |
| MON_GET_INDEX table function - Get index metrics  | ACTIVITY METRICS BASE            |
| MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE table function - Get table metrics  | ACTIVITY METRICS BASE            |
| MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list   | ACTIVITY METRICS BASE            |

Table 2219. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 2220. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch   |
|------------------------|-----------------------|------------------|
| Locking                | -                     | Always collected |
| Tables                 | event_table           | Always collected |
| Deadlocks <sup>1</sup> | lock                  | Always collected |
| Deadlocks <sup>1</sup> | event_dlconn          | Always collected |

Table 2220. Event Monitoring Information (continued)

| Event Type                          | Logical Data Grouping | Monitor Switch   |
|-------------------------------------|-----------------------|------------------|
| Deadlocks with Details <sup>1</sup> | event_detailed_dlconn | Always collected |

1 This event monitor has been deprecated. Its use is no longer recommended and might be removed in a future release. Use the CREATE EVENT MONITOR FOR LOCKING statement to monitor lock-related events, such as lock timeouts, lock waits, and deadlocks.

## Usage

Along with **table\_name**, this element can help you determine the source of contention for resources.

For application-level, application-lock-level, deadlock-monitoring-level, this is the schema of the table that the application is waiting to lock, because it is currently locked by another application. This element is only set if **lock\_object\_type** indicates that the application is waiting to obtain a table lock. For snapshot monitoring at the application-level and application-lock levels, this item is only valid when the “lock” monitor group information is set to ON.

For snapshot monitoring at the object-lock level, this item is returned for table and row level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this element identifies the schema of the table for which information has been collected. For temporary tables, the format for **table\_schema** is “<agent\_id><auth\_id>”, where:

- *agent\_id* is the Application Handle of the application creating the temporary table
- *auth\_id* is the authorization ID used by the application to connect to the database

---

## tablespace\_min\_recovery\_time - Minimum recovery time for rollforward monitor element

A timestamp showing the earliest point in time to which a table space can be rolled forward. The timestamp reflects local time.

Table 2221. Table Function Monitoring Information

| Table Function  | Monitor Element Collection Level |
|---|----------------------------------|
| MON_GET_TABLESPACE table function - Get table space metrics | ACTIVITY METRICS BASE            |

Table 2222. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

## Usage

Displayed only if non zero.

---

## target\_cf\_gbp\_size - Target cluster caching facility group buffer pool size monitor element

During a dynamic resize, this monitor element shows the group buffer pool memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 2223. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## target\_cf\_lock\_size - Target cluster caching facility lock size monitor element

During a dynamic resize, this monitor element shows the global lock memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 2224. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |

---

## target\_cf\_sca\_size - Target cluster caching facility shared communications area size monitor element

During a dynamic resize, this monitor element shows the shared communications area memory target value, in pages with a page size of 4 KB. A resize is complete when the target value matches the configured value.

*Table 2225. Table Function Monitoring Information*

| Table Function                             | Monitor Element Collection Level |
|--|----------------------------------|
| MON_GET_CF table function - Get CF metrics | ACTIVITY METRICS BASE            |



---

## Part 4. Appendixes



---

## Appendix A. Overview of the DB2 technical information

DB2 technical information is available in multiple formats that can be accessed in multiple ways.

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command-line help
  - Command help
  - Message help

**Note:** The DB2 Information Center topics are updated more frequently than either the PDF or the hardcopy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at [ibm.com](http://ibm.com).

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at [ibm.com](http://ibm.com). Access the DB2 Information Management software library site at <http://www.ibm.com/software/data/sw-library/>.

### Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to [db2docs@ca.ibm.com](mailto:db2docs@ca.ibm.com). The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this email address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

---

## DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at [www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss). English and translated DB2 Version 10.1 manuals in PDF format can be downloaded from [www.ibm.com/support/docview.wss?rs=71&uid=swg2700947](http://www.ibm.com/support/docview.wss?rs=71&uid=swg2700947).

Although the tables identify books available in print, the books might not be available in your country or region.

The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed below.

**Note:** The *DB2 Information Center* is updated more frequently than either the PDF or the hard-copy books.

*Table 2226. DB2 technical information*

| <b>Name</b>   | <b>Form Number</b> | <b>Available in print</b> | <b>Last updated</b> |
|---|--------------------|---------------------------|---------------------|
| <i>Administrative API Reference</i>                                 | SC27-3864-00       | Yes                       | April, 2012         |
| <i>Administrative Routines and Views</i>                            | SC27-3865-00       | No                        | April, 2012         |
| <i>Call Level Interface Guide and Reference Volume 1</i>            | SC27-3866-00       | Yes                       | April, 2012         |
| <i>Call Level Interface Guide and Reference Volume 2</i>            | SC27-3867-00       | Yes                       | April, 2012         |
| <i>Command Reference</i>  | SC27-3868-00       | Yes                       | April, 2012         |
| <i>Database Administration Concepts and Configuration Reference</i> | SC27-3871-00       | Yes                       | April, 2012         |
| <i>Data Movement Utilities Guide and Reference</i>                  | SC27-3869-00       | Yes                       | April, 2012         |
| <i>Database Monitoring Guide and Reference</i>                      | SC27-3887-00       | Yes                       | April, 2012         |
| <i>Data Recovery and High Availability Guide and Reference</i>      | SC27-3870-00       | Yes                       | April, 2012         |
| <i>Database Security Guide</i>                                      | SC27-3872-00       | Yes                       | April, 2012         |
| <i>DB2 Workload Management Guide and Reference</i>                  | SC27-3891-00       | Yes                       | April, 2012         |
| <i>Developing ADO.NET and OLE DB Applications</i>                   | SC27-3873-00       | Yes                       | April, 2012         |
| <i>Developing Embedded SQL Applications</i>                         | SC27-3874-00       | Yes                       | April, 2012         |
| <i>Developing Java Applications</i>                                 | SC27-3875-00       | Yes                       | April, 2012         |
| <i>Developing Perl, PHP, Python, and Ruby on Rails Applications</i> | SC27-3876-00       | No                        | April, 2012         |
| <i>Developing User-defined Routines (SQL and External)</i>          | SC27-3877-00       | Yes                       | April, 2012         |
| <i>Getting Started with Database Application Development</i>        | GI13-2046-00       | Yes                       | April, 2012         |

Table 2226. DB2 technical information (continued)

| <b>Name</b>  | <b>Form Number</b> | <b>Available in print</b> | <b>Last updated</b> |
|--|--------------------|---------------------------|---------------------|
| <i>Getting Started with DB2 Installation and Administration on Linux and Windows</i> | GI13-2047-00       | Yes                       | April, 2012         |
| <i>Globalization Guide</i>   | SC27-3878-00       | Yes                       | April, 2012         |
| <i>Installing DB2 Servers</i>  | GC27-3884-00       | Yes                       | April, 2012         |
| <i>Installing IBM Data Server Clients</i>  | GC27-3883-00       | No                        | April, 2012         |
| <i>Message Reference Volume 1</i>  | SC27-3879-00       | No                        | April, 2012         |
| <i>Message Reference Volume 2</i>  | SC27-3880-00       | No                        | April, 2012         |
| <i>Net Search Extender Administration and User's Guide</i>                           | SC27-3895-00       | No                        | April, 2012         |
| <i>Partitioning and Clustering Guide</i>   | SC27-3882-00       | Yes                       | April, 2012         |
| <i>pureXML Guide</i>   | SC27-3892-00       | Yes                       | April, 2012         |
| <i>Spatial Extender User's Guide and Reference</i>                                   | SC27-3894-00       | No                        | April, 2012         |
| <i>SQL Procedural Languages: Application Enablement and Support</i>                  | SC27-3896-00       | Yes                       | April, 2012         |
| <i>SQL Reference Volume 1</i>  | SC27-3885-00       | Yes                       | April, 2012         |
| <i>SQL Reference Volume 2</i>  | SC27-3886-00       | Yes                       | April, 2012         |
| <i>Text Search Guide</i>   | SC27-3888-00       | Yes                       | April, 2012         |
| <i>Troubleshooting and Tuning Database Performance</i>                               | SC27-3889-00       | Yes                       | April, 2012         |
| <i>Upgrading to DB2 Version 10.1</i>   | SC27-3881-00       | Yes                       | April, 2012         |
| <i>What's New for DB2 Version 10.1</i>   | SC27-3890-00       | Yes                       | April, 2012         |
| <i>XQuery Reference</i>  | SC27-3893-00       | No                        | April, 2012         |

Table 2227. DB2 Connect-specific technical information

| <b>Name</b>  | <b>Form Number</b> | <b>Available in print</b> | <b>Last updated</b> |
|--|--------------------|---------------------------|---------------------|
| <i>DB2 Connect Installing and Configuring DB2 Connect Personal Edition</i> | SC27-3861-00       | Yes                       | April, 2012         |
| <i>DB2 Connect Installing and Configuring DB2 Connect Servers</i>          | SC27-3862-00       | Yes                       | April, 2012         |
| <i>DB2 Connect User's Guide</i>  | SC27-3863-00       | Yes                       | April, 2012         |

---

## Displaying SQL state help from the command line processor

DB2 products return an SQLSTATE value for conditions that can be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

### Procedure

To start SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, ? 08003 displays help for the 08003 SQL state, and ? 08 displays help for the 08 class code.

---

## Accessing different versions of the DB2 Information Center

Documentation for other versions of DB2 products is found in separate information centers on [ibm.com](http://ibm.com)<sup>®</sup>.

### About this task

For DB2 Version 10.1 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1>.

For DB2 Version 9.8 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/>.

For DB2 Version 9.7 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/>.

For DB2 Version 9.5 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>.

For DB2 Version 9.1 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>.

For DB2 Version 8 topics, go to the *DB2 Information Center* URL at: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

---

## Updating the DB2 Information Center installed on your computer or intranet server

A locally installed DB2 Information Center must be updated periodically.

### Before you begin

A DB2 Version 10.1 Information Center must already be installed. For details, see the “Installing the DB2 Information Center using the DB2 Setup wizard” topic in *Installing DB2 Servers*. All prerequisites and restrictions that applied to installing the Information Center also apply to updating the Information Center.

## About this task

An existing DB2 Information Center can be updated automatically or manually:

- Automatic updates update existing Information Center features and languages. One benefit of automatic updates is that the Information Center is unavailable for a shorter time compared to during a manual update. In addition, automatic updates can be set to run as part of other batch jobs that run periodically.
- Manual updates can be used to update existing Information Center features and languages. Automatic updates reduce the downtime during the update process, however you must use the manual process when you want to add features or languages. For example, a local Information Center was originally installed with both English and French languages, and now you want to also install the German language; a manual update will install German, as well as, update the existing Information Center features and languages. However, a manual update requires you to manually stop, update, and restart the Information Center. The Information Center is unavailable during the entire update process. In the automatic update process the Information Center incurs an outage to restart the Information Center after the update only.

This topic details the process for automatic updates. For manual update instructions, see the “Manually updating the DB2 Information Center installed on your computer or intranet server” topic.

## Procedure

To automatically update the DB2 Information Center installed on your computer or intranet server:

1. On Linux operating systems,
  - a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the `/opt/ibm/db2ic/V10.1` directory.
  - b. Navigate from the installation directory to the `doc/bin` directory.
  - c. Run the `update-ic` script:

```
update-ic
```
2. On Windows operating systems,
  - a. Open a command window.
  - b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the `<Program Files>\IBM\DB2 Information Center\Version 10.1` directory, where `<Program Files>` represents the location of the Program Files directory.
  - c. Navigate from the installation directory to the `doc\bin` directory.
  - d. Run the `update-ic.bat` file:

```
update-ic.bat
```

## Results

The DB2 Information Center restarts automatically. If updates were available, the Information Center displays the new and updated topics. If Information Center updates were not available, a message is added to the log. The log file is located in `doc\eclipse\configuration` directory. The log file name is a randomly generated number. For example, `1239053440785.log`.

---

## Manually updating the DB2 Information Center installed on your computer or intranet server

If you have installed the DB2 Information Center locally, you can obtain and install documentation updates from IBM.

### About this task

Updating your locally installed *DB2 Information Center* manually requires that you:

1. Stop the *DB2 Information Center* on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to apply updates. The Workstation version of the DB2 Information Center always runs in stand-alone mode. .
2. Use the Update feature to see what updates are available. If there are updates that you must install, you can use the Update feature to obtain and install them

**Note:** If your environment requires installing the *DB2 Information Center* updates on a machine that is not connected to the internet, mirror the update site to a local file system by using a machine that is connected to the internet and has the *DB2 Information Center* installed. If many users on your network will be installing the documentation updates, you can reduce the time required for individuals to perform the updates by also mirroring the update site locally and creating a proxy for the update site.

If update packages are available, use the Update feature to get the packages. However, the Update feature is only available in stand-alone mode.

3. Stop the stand-alone Information Center, and restart the *DB2 Information Center* on your computer.

**Note:** On Windows 2008, Windows Vista (and higher), the commands listed later in this section must be run as an administrator. To open a command prompt or graphical tool with full administrator privileges, right-click the shortcut and then select **Run as administrator**.

### Procedure

To update the *DB2 Information Center* installed on your computer or intranet server:

1. Stop the *DB2 Information Center*.
  - On Windows, click **Start > Control Panel > Administrative Tools > Services**. Then right-click **DB2 Information Center** service and select **Stop**.
  - On Linux, enter the following command:  

```
/etc/init.d/db2icdv10 stop
```
2. Start the Information Center in stand-alone mode.
  - On Windows:
    - a. Open a command window.
    - b. Navigate to the path where the Information Center is installed. By default, the *DB2 Information Center* is installed in the *Program\_Files\IBM\DB2 Information Center\Version 10.1* directory, where *Program\_Files* represents the location of the Program Files directory.
    - c. Navigate from the installation directory to the `doc\bin` directory.
    - d. Run the `help_start.bat` file:

```
help_start.bat
```

- On Linux:
  - a. Navigate to the path where the Information Center is installed. By default, the *DB2 Information Center* is installed in the `/opt/ibm/db2ic/V10.1` directory.
  - b. Navigate from the installation directory to the `doc/bin` directory.
  - c. Run the `help_start` script:

```
help_start
```

The systems default Web browser opens to display the stand-alone Information Center.

3. Click the **Update** button (🔧). (JavaScript must be enabled in your browser.) On the right panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the installation process, check that the selections you want to install, then click **Install Updates**.
5. After the installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center:
  - On Windows, navigate to the `doc\bin` directory within the installation directory, and run the `help_end.bat` file:

```
help_end.bat
```

**Note:** The `help_end` batch file contains the commands required to safely stop the processes that were started with the `help_start` batch file. Do not use `Ctrl-C` or any other method to stop `help_start.bat`.
  - On Linux, navigate to the `doc/bin` directory within the installation directory, and run the `help_end` script:

```
help_end
```

**Note:** The `help_end` script contains the commands required to safely stop the processes that were started with the `help_start` script. Do not use any other method to stop the `help_start` script.
7. Restart the *DB2 Information Center*.
  - On Windows, click **Start > Control Panel > Administrative Tools > Services**. Then right-click **DB2 Information Center** service and select **Start**.
  - On Linux, enter the following command:

```
/etc/init.d/db2icdv10 start
```

## Results

The updated *DB2 Information Center* displays the new and updated topics.

---

## DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 database products. Lessons provide step-by-step instructions.

### Before you begin

You can view the XHTML version of the tutorial from the Information Center at <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/>.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

## DB2 tutorials

To view the tutorial, click the title.

### “pureXML” in *pureXML Guide*

Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

---

## DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 database products.

### DB2 documentation

Troubleshooting information can be found in the *Troubleshooting and Tuning Database Performance* or the Database fundamentals section of the *DB2 Information Center*, which contains:

- Information about how to isolate and identify problems with DB2 diagnostic tools and utilities.
- Solutions to some of the most common problem.
- Advice to help solve other problems you might encounter with your DB2 database products.

### IBM Support Portal

See the IBM Support Portal if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the IBM Support Portal at [http://www.ibm.com/support/entry/portal/Overview/Software/Information\\_Management/DB2\\_for\\_Linux,\\_UNIX\\_and\\_Windows](http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows)

---

## Terms and conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

**Applicability:** These terms and conditions are in addition to any terms of use for the IBM website.

**Personal use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

**Commercial use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights:** Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

**IBM Trademarks:** IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)



---

## Appendix B. Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements, changes, or both in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by IBM are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licenses of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited  
U59/3600  
3600 Steeles Avenue East  
Markham, Ontario L3R 9Z7  
CANADA

Such information may be available, subject to appropriate terms and conditions, including, in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *\_enter the year or years\_*. All rights reserved.

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, its affiliates, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



---

# Index

## Special characters

\_DETAILS table functions 13  
.db2top configuration file 436  
.db2toprc configuration file 436

## A

### activities

associated with statements 270

#### monitor elements

act\_aborted\_total 637  
act\_completed\_total 638  
act\_rejected\_total 640  
act\_throughput 643  
act\_total 643  
activity\_collected 645  
activity\_id 645  
activity\_secondary\_id 646  
activity\_state 647  
activity\_type 647  
coord\_act\_aborted\_total 742  
coord\_act\_completed\_total 742  
coord\_act\_rejected\_total 748  
parent\_activity\_id 1014

#### monitoring 521

### activities monitor elements

acc\_curs\_blk 636  
act\_exec\_time 639  
act\_remapped\_in  
  details 641  
act\_remapped\_out  
  details 641  
act\_rqsts\_total 642  
activation time  
  last\_wlm\_reset 898  
active\_sorts 644

### activities

act\_aborted\_total 637  
act\_completed\_total 638  
act\_rejected\_total 640  
act\_throughput 643  
act\_total 643  
activity\_collected 645  
activity\_id 645  
activity\_secondary\_id 646  
activity\_state 647  
activity\_type 647  
coord\_act\_aborted\_total 742  
coord\_act\_completed\_total 742  
coord\_act\_rejected\_total 748  
parent\_activity\_id 1014

### ACTIVITYTOTALTIME activity threshold

activitytotaltime\_threshold\_id 648  
activitytotaltime\_threshold\_value 648  
activitytotaltime\_threshold\_violated 649

adapter\_name 649

address 649

agent\_tid 653

### agents

agent\_id 650

### activities monitor elements (*continued*)

#### agents (*continued*)

agent\_id\_holding\_lock 651  
agent\_pid 652  
agent\_status 652  
agent\_sys\_cpu\_time 653  
agent\_usr\_cpu\_time 654  
agent\_wait\_time 654  
agent\_waits\_total 655  
agents\_created\_empty\_pool 656  
agents\_from\_pool 657  
agents\_registered 657  
agents\_registered\_top 658  
agents\_stolen 658  
agents\_top 658  
agents\_waiting\_on\_token 659  
agents\_waiting\_top 659  
appl\_priority 671  
associated\_agents\_top 678  
coord\_agent\_pid 749  
coord\_agents\_top 749  
idle\_agents 873  
max\_agent\_overflows 943  
num\_agents 972  
num\_assoc\_agents 972  
priv\_workspace\_size\_top 1159  
quiescer\_agent\_id 1167  
rolled\_back\_agent\_id 1186

agg\_temp\_tablespace\_top 660

aggsqltempspace\_threshold\_value 661

aggsqltempspace\_threshold\_violated 661

### aliases

client\_db\_alias 711  
input\_db\_alias 878

app\_act\_aborted\_total 661

app\_act\_completed\_total 662

app\_act\_rejected\_total 664

appl\_action 665

### applications

appl\_id 666  
appl\_id\_holding\_lk 668  
appl\_id\_oldest\_xact 669  
appl\_idle\_time 670  
appl\_name 670  
appl\_priority\_type 672  
appl\_section\_inserts 672  
appl\_section\_lookups 673  
appl\_status 673  
application\_handle 675  
client\_applname 710  
memory\_pool\_used 964  
tpmon\_client\_app 1389

async\_read\_time 678

async\_write\_time 678

### attributes

progress\_list\_attr 1161  
audit\_subsystem\_wait\_time 683  
audit\_subsystem\_waits\_total 685

### audits

audit\_events\_total 679  
audit\_file\_write\_wait\_time 680

activities monitor elements (*continued*)

- audits (*continued*)
  - audit\_file\_writes\_total 682
- auth\_id 686
- authority\_bitmap 687
- authorization IDs
  - execution\_id 821
  - session\_auth\_id 1212
- auto\_storage\_hybrid 689
- automatic storage path
  - sto\_path\_free\_sz 1263
- binds\_precompiles 690
- blocking\_cursor 691
- blocks\_pending\_cleanup 692
- boundary\_leaf\_node\_splits 693
- buffer pools 449
  - automatic 689
  - block\_ios 690
  - bp\_cur\_buffsz 693
  - bp\_id 693
  - bp\_name 693
  - bp\_new\_buffsz 694
  - bp\_pages\_left\_to\_remove 694
  - bp\_tbsp\_use\_count 694
  - buff\_free 695
  - buff\_free\_bottom 695
  - DB2 pureScale environments 1465
  - object\_data\_l\_reads 987
  - object\_data\_p\_reads 988
  - object\_index\_l\_reads 991
  - object\_index\_p\_reads 992
  - object\_xda\_l\_reads 996
  - object\_xda\_p\_reads 997
  - pool\_async\_data\_read\_reqs 1027
  - pool\_async\_data\_reads 1028
  - pool\_async\_data\_writes 1029
  - pool\_async\_index\_read\_reqs 1032
  - pool\_async\_index\_reads 1033
  - pool\_async\_index\_writes 1034
  - pool\_async\_read\_time 1034
  - pool\_async\_write\_time 1035
  - pool\_async\_xda\_gbp\_invalid\_pages 1037, 1516
  - pool\_async\_xda\_gbp\_l\_reads 1037, 1516
  - pool\_async\_xda\_gbp\_p\_reads 1038, 1517
  - pool\_async\_xda\_lbp\_pages\_found 1038, 1517
  - pool\_async\_xda\_read\_reqs 1039
  - pool\_async\_xda\_reads 1039
  - pool\_async\_xda\_writes 1040
  - pool\_data\_l\_reads 1049
  - pool\_data\_p\_reads 1051
  - pool\_data\_writes 1053
  - pool\_drty\_pg\_steal\_clns 1056
  - pool\_drty\_pg\_thrsh\_clns 1057
  - pool\_index\_l\_reads 1081
  - pool\_index\_p\_reads 1083
  - pool\_index\_writes 1085
  - pool\_lsn\_gap\_clns 1087
  - pool\_no\_victim\_buffer 1088
  - pool\_read\_time 1114
  - pool\_temp\_data\_l\_reads 1117
  - pool\_temp\_data\_p\_reads 1119
  - pool\_temp\_index\_l\_reads 1121
  - pool\_temp\_index\_p\_reads 1122
  - pool\_temp\_xda\_l\_reads 1124
  - pool\_temp\_xda\_p\_reads 1126
  - pool\_write\_time 1129
  - pool\_xda\_gbp\_invalid\_pages 1132, 1530

activities monitor elements (*continued*)

- buffer pools (*continued*)
  - pool\_xda\_gbp\_l\_reads 1133, 1531
  - pool\_xda\_gbp\_p\_reads 1135, 1533
  - pool\_xda\_l\_reads 1137
  - pool\_xda\_lbp\_pages\_found 1139, 1534
  - pool\_xda\_p\_reads 1140
  - pool\_xda\_writes 1142
- buffers
  - num\_log\_data\_found\_in\_buffer 978
- byte order
  - byte\_order 697
- caches
  - stats\_cache\_size 1243
- cat\_cache\_inserts 698
- cat\_cache\_lookups 699
- cat\_cache\_overflows 700
- cat\_cache\_size\_top 701
- catalog\_node 702
- catalog\_node\_name 702
- cf\_wait\_time 703, 1489
- cf\_waits 703, 1489
- change history
  - backup\_timestamp 689
  - cfg\_collection\_type 704
  - cfg\_name 704
  - cfg\_old\_value 705
  - cfg\_old\_value\_flags 705
  - cfg\_value 706
  - cfg\_value\_flags 706
  - ddl\_classification 781
  - deferred 786
  - device\_type 788
  - location 901
  - location\_type 902
  - phase\_start\_event\_id 1019
  - phase\_start\_event\_timestamp 1019
  - regvar\_collection\_type 1173
  - regvar\_level 1173
  - regvar\_name 1173
  - regvar\_old\_value 1174
  - regvar\_value 1174
  - savepoint\_id 1199
  - start\_event\_id 1241
  - start\_event\_timestamp 1241
  - tbsp\_names 1298
  - txn\_completion\_status 1399
  - utility\_detail 1413
  - utility\_invocation\_id 1414
  - utility\_operation\_type 1415
  - utility\_phase\_detail 1416
  - utility\_phase\_type 1416
  - utility\_start\_type 1417
  - utility\_stop\_type 1418
- client\_hostname 711
- client\_nname 713
- client\_pid 713
- client\_platform 714
- client\_port\_number 715
- client\_prdid 715
- code pages
  - codepage\_id 718
  - host\_ccsid 870
- collection levels 525
- comm\_exit\_wait\_time 719
- comm\_exit\_waits 720
- comm\_private\_mem 721

activities monitor elements (*continued*)

commit\_sql\_stmts 721  
 commits  
   int\_commits 881  
 communication protocols  
   client\_protocol 716  
 CONCURRENTDBCOORDACTIVITIES threshold  
   concurrentdbcoordactivities\_wl\_was  
     \_threshold\_id 733  
   concurrentdbcoordactivities\_wl\_was  
     \_threshold\_queued 733  
   concurrentdbcoordactivities\_wl\_was  
     \_threshold\_value 733  
   concurrentdbcoordactivities\_wl\_was  
     \_threshold\_violated 734  
   concurrentdbcoordactivities\_db\_threshold\_id 727  
   concurrentdbcoordactivities\_subclass\_  
     \_threshold\_value 730  
   concurrentdbcoordactivities\_subclass\_  
     \_threshold\_queued 729  
   concurrentdbcoordactivities\_subclass\_  
     \_threshold\_violated 730  
   concurrentdbcoordactivities\_superclass\_  
     \_threshold\_id 731  
   concurrentdbcoordactivities\_superclass\_  
     \_threshold\_queued 731  
   concurrentdbcoordactivities\_superclass\_  
     \_threshold\_value 732  
   concurrentdbcoordactivities\_superclass\_  
     \_threshold\_violated 732  
   concurrentdbcoordactivities\_work\_action\_set  
     \_threshold\_queued 735  
   concurrentdbcoordactivities\_work\_action\_set  
     \_threshold\_value 735  
   concurrentdbcoordactivities\_work\_action\_set  
     \_threshold\_violated 736  
   concurrentdbcoordactivities\_work\_action\_  
     \_set\_threshold\_id 734  
 configured\_cf\_gbp\_size 723, 1490  
 configured\_cf\_lock\_size 723, 1490  
 configured\_cf\_mem\_size 724, 1491  
 configured\_cf\_sca\_size 723, 1491  
 connection\_start\_time 737  
 connections  
   appl\_con\_time 666  
   appls\_cur\_cons 677  
   appls\_in\_db2 677  
   con\_elapsed\_time 724  
   con\_local\_dbases 724  
   conn\_complete\_time 736  
   conn\_time 736  
   connection\_status 737  
   connections\_top 738  
   gw\_connections\_top 854  
   gw\_cons\_wait\_client 854  
   gw\_cons\_wait\_host 854  
   gw\_cur\_cons 855  
   gw\_total\_cons 856  
   local\_cons 900  
   local\_cons\_in\_exec 900  
   num\_gw\_conn\_switches 975  
   rem\_cons\_in 1175  
   rem\_cons\_in\_exec 1175  
   total\_cons 1326  
   total\_sec\_cons 1367  
 containers  
   container\_accessible 739

activities monitor elements (*continued*)

containers (*continued*)  
   container\_id 739  
   container\_name 739  
   container\_total\_pages 740  
   container\_type 741  
   container\_usable\_pages 741  
 coord\_act\_est\_cost\_avg 743  
 coord\_act\_exec\_time\_avg 744  
 coord\_act\_interarrival\_time\_avg 745  
 coord\_act\_lifetime\_avg 745  
 coord\_act\_queue\_time\_avg 747  
 coord\_agent\_tid 747  
 coord\_member 750  
 country\_code  
   see monitor elements, territory\_code 1305  
 CPU time  
   ss\_sys\_cpu\_time 1240  
   ss\_usr\_cpu\_time 1240  
   stmt\_sys\_cpu\_time 1256  
   stmt\_usr\_cpu\_time 1259  
   system\_cpu\_time 1268  
   total\_cpu\_time 1332  
   total\_sys\_cpu\_time 1385  
   total\_usr\_cpu\_time 1387  
   user\_cpu\_time 1412  
 cpu\_configured 753  
 cpu\_cores\_per\_socket 754  
 cpu\_hmt\_degree 754  
 cpu\_idle 754  
 cpu\_iowait 755  
 cpu\_load\_long 756  
 cpu\_load\_medium 756  
 cpu\_load\_short 756  
 cpu\_online 757  
 cpu\_speed 757  
 cpu\_system 757  
 cpu\_timebase 758  
 cpu\_total 758  
 cpu\_usage\_total 759  
 cpu\_user 759  
 cputime\_threshold\_id 761  
 cputime\_threshold\_value 762  
 cputime\_threshold\_violated 762  
 cputimeinsc\_threshold\_id 762  
 cputimeinsc\_threshold\_value 763  
 cputimeinsc\_threshold\_violated 763  
 current\_cf\_gbp\_size 765, 1491  
 current\_cf\_lock\_size 765, 1491  
 current\_cf\_mem\_size 765, 1492  
 current\_cf\_sca\_size 765, 1492  
 current\_request 767  
 cursors  
   cursor\_name 767  
   rej\_curs\_blk 1174  
 data organization 444  
 data\_object\_l\_pages - Table data logical pages 768  
 database manager  
   server\_db2\_type 1206  
 database paths  
   db\_path 775  
 datataginsc\_threshold\_id 770  
 datataginsc\_threshold\_value 770  
 datataginsc\_threshold\_violated 770  
 datatagnotinsc\_threshold\_id 771  
 datatagnotinsc\_threshold\_value 771  
 datatagnotinsc\_threshold\_violated 771

activities monitor elements (*continued*)

- db\_heap\_top 774
- db\_storage\_path 776
- db\_storage\_path\_id 777
- DB2 Connect
  - gw\_con\_time 853
  - gw\_exec\_time 855
- db2\_process\_identifier 772
- db2\_process\_name 772
- dbpartitionnum 779, 1493
- deadlock\_member 783
- deadlock\_type 784
- deadlocks
  - deadlock\_id 783
  - deadlock\_node 784
  - deadlocks 784
  - dl\_conns 805
  - int\_deadlock\_rollbacks 882
- del\_keys\_cleaned 787
- DELETE statement
  - delete\_sql\_stmts 787
- descriptors
  - progress\_description 1161
- destination\_service\_class\_id 788
- disabled\_peds 803
- edu\_id 806
- eff\_stmt\_text 806
- effective\_query\_degree 807
- empty\_pages\_deleted 808
- empty\_pages\_reused 808
- entry\_time 809
- environment handles
  - comp\_env\_desc 722
- errors
  - gw\_comm\_errors 853
- estimatedsqlcost\_threshold\_id 809
- estimatedsqlcost\_threshold\_value 810
- estimatedsqlcost\_threshold\_violated 810
- event monitors 40, 547
  - count 753
  - event\_monitor\_name 812
  - evmon\_activates 814
  - evmon\_flushes 820
- event\_id 811
- event\_timestamp 812
- event\_type 813
- events
  - event\_time 812
  - start\_time 1242
  - stop\_time 1263
- executable\_id 819, 821
- executable\_list\_size 820
- executable\_list\_truncated 820
- fabrications
  - stats\_fabricate\_time 1244
  - stats\_fabrications 1245
- fast communication manager (FCM)
  - buff\_auto\_tuning 695
  - ch\_auto\_tuning 707
- fast communication manager (FCM)
  - buff\_max 696
  - buff\_total 696
  - ch\_free 707
  - ch\_free\_bottom 707
  - ch\_max 708
  - ch\_total 708
  - fcm\_message\_rcv\_volume 824

activities monitor elements (*continued*)

- fast communication manager (FCM) (*continued*)
  - fcm\_message\_rcv\_wait\_time 825
  - hostname 871
  - remote\_member 1176
  - total\_buffers\_rcvd 1319
  - total\_buffers\_sent 1320
- fcm\_congested\_sends 823
- fcm\_congestion\_time 823
- fcm\_message\_rcvs\_total 827
- fcm\_message\_sends\_total 831
- fcm\_num\_congestion\_timeouts 823
- fcm\_num\_conn\_lost 823
- fcm\_num\_conn\_timeouts 824
- fcm\_rcv\_volume 832
- fcm\_rcv\_wait\_time 833
- fcm\_rcvs\_total 835
- fcm\_send\_volume 836
- fcm\_send\_wait\_time 837
- fcm\_sends\_total 839
- fcm\_tq\_rcv\_volume 840
- fcm\_tq\_rcv\_wait\_time 841
- fcm\_tq\_rcvs\_total 843
- fcm\_tq\_send\_volume 844
- fcm\_tq\_send\_wait\_time 845
- fcm\_tq\_sends\_total 847
- federated servers
  - disconnects 804
- fetching
  - fetch\_count 848
- file systems
  - fs\_caching 850
  - fs\_id 851
  - fs\_total\_size 851
  - fs\_used\_size 852
- files
  - files\_closed 849
- global variables
  - mon\_interval\_id 968
- global\_transaction\_id 852
- group buffer pools
  - object\_data\_gbp\_l\_reads 985
  - object\_data\_gbp\_p\_reads 986
  - object\_index\_gbp\_invalid\_pages 989
  - object\_index\_gbp\_l\_reads 989
  - object\_index\_gbp\_p\_reads 990
  - object\_xda\_gbp\_invalid\_pages 994
  - object\_xda\_gbp\_l\_reads 995
  - object\_xda\_gbp\_p\_reads 995
- gw\_comm\_error\_time 853
- hash joins
  - active\_hash\_joins 644
  - hash\_join\_overflows 868
  - hash\_join\_small\_overflows 868
  - post\_shrthreshold\_hash\_joins 1144
  - post\_threshold\_hash\_joins 1146
  - total\_hash\_joins 1339
- high availability disaster recovery (HADR)
  - hadr\_connect\_status 856
  - hadr\_connect\_time 857
  - hadr\_heartbeat 858
  - hadr\_local\_host 858
  - hadr\_local\_service 859
  - hadr\_log\_gap 859
  - hadr\_peer\_window 860
  - hadr\_peer\_window\_end 860
  - hadr\_primary\_log\_file 861

activities monitor elements (*continued*)

- high availability disaster recovery (HADR) (*continued*)
  - hadr\_primary\_log\_lsn 861
  - hadr\_primary\_log\_page 862
  - hadr\_remote\_host 862
  - hadr\_remote\_instance 863
  - hadr\_remote\_service 863
  - hadr\_role 864
  - hadr\_standby\_log\_file 864
  - hadr\_standby\_log\_lsn 865
  - hadr\_standby\_log\_page 865
  - hadr\_state 865
  - hadr\_syncmode 866
  - hadr\_timeout 867
- histograms
  - histogram\_type 869
  - number\_in\_bin 984
  - top 1312
- host databases
  - host\_db\_name 871
- host\_name 872, 1495
- I/O
  - num\_log\_part\_page\_io 978
  - num\_log\_read\_io 979
  - num\_log\_write\_io 979
  - num\_pages\_from\_block\_IOs 1012
  - num\_pages\_from\_vectored\_IOs 1013
  - vectored\_ios 1420
- id 873, 1495
- identifiers
  - arm\_correlator 677
  - bin\_id 690
  - db\_work\_action\_set\_id 778
  - db\_work\_class\_id 779
  - host\_prdid 872
  - sc\_work\_action\_set\_id 1199
  - sc\_work\_class\_id 1200
  - service\_class\_id 1208
  - sql\_req\_id 1229
  - work\_action\_set\_id 1425
  - work\_class\_id 1426
- inbound\_bytes\_received 874
- inbound\_bytes\_sent 874
- inbound\_comm\_address 875
- include\_col\_updates 875
- incremental\_bind 875
- index\_jump\_scans 875
- index\_name 876
- index\_object\_l\_pages - Index data logical pages 877
- index\_schema 876
- indexes
  - iid 874
  - index\_name 876
  - index\_object\_pages 876
  - index\_only\_scans 877
  - index\_scans 877
  - index\_schema 876
  - index\_tbsp\_id 878
  - int\_node\_splits 883
  - nleaf 970
  - nlevels 971
  - page\_allocations 1010
  - pages\_merged 1013
  - root\_node\_splits 1188
- insert\_timestamp 879
- int\_rows\_deleted 885
- interfaces for viewing metrics in XML documents 18

activities monitor elements (*continued*)

- interfaces that return XML documents 13
- intra\_parallel\_state 887
- ipc\_send\_wait\_time 892
- is\_system\_appl 894
- isolation levels
  - effective\_isolation 806
- key\_updates 894
- large objects (LOBs)
  - lob\_object\_pages 899
- last\_executable\_id 895
- last\_extent 895
- last\_reference\_time 896
- last\_request\_type 896
- last\_updated 898
- lob\_object\_l\_pages - LOB data logical pages 899
- local buffer pools
  - object\_data\_lbp\_pages\_found 987
  - object\_index\_lbp\_pages\_found 990
  - object\_xda\_lbp\_pages\_found 996
- local\_transaction\_id 901
- locations
  - db\_location 774
- lock modes
  - lock\_current\_mode 904
  - lock\_mode 913
  - lock\_mode\_requested 914
- lock\_escals\_global 908, 1498
- lock\_escals\_locklist 910, 1499
- lock\_escals\_maxlocks 911, 1500
- lock\_timeouts\_global 923, 1502
- lock\_wait\_end\_time 924
- lock\_wait\_time\_global 927, 1503
- lock\_wait\_time\_global\_top 929, 1504
- lock\_wait\_val 929
- lock\_waits\_global 931, 1505
- locking
  - hld\_application\_handle 870
  - hld\_member 870
  - req\_agent\_tid 1183
  - req\_application\_handle 1183
  - req\_executable\_id 1183
  - req\_member 1183
- locks
  - DB2 pureScale environments 1477
  - effective\_lock\_timeout 807
  - lock\_attributes 902
  - lock\_count 903
  - lock\_escals 906, 1496
  - lock\_hold\_count 912
  - lock\_list\_in\_use 913
  - lock\_name 915
  - lock\_node 916
  - lock\_object\_name 916
  - lock\_object\_type 917
  - lock\_release\_flags 919
  - lock\_status 920
  - lock\_timeout\_val 921
  - lock\_timeouts 921
  - lock\_wait\_time 925
  - lock\_waits 929
  - locks\_held 933
  - locks\_held\_top 933
  - locks\_in\_list 934
  - locks\_waiting 934
  - participant\_no\_holding\_lk 1016
  - remote\_lock\_time 1176

activities monitor elements (*continued*)

- locks (*continued*)
  - remote\_locks 1176
  - sequence\_no\_holding\_lk 1205
  - stmt\_lock\_timeout 1250
  - uow\_lock\_wait\_time 1404
  - x\_lock\_escals 1430
- log buffers
  - num\_log\_buffer\_full 976
- log files
  - current\_active\_log 766
  - current\_archive\_log 766
  - diaglog\_write\_wait\_time 789
  - diaglog\_writes\_total 790
  - first\_active\_log 849
  - last\_active\_log 894
  - log\_read\_time 939
  - log\_reads 939
  - sec\_logs\_allocated 1201
- log space
  - log\_held\_by\_dirty\_pages 938
  - log\_to\_redo\_for\_recovery 940
  - log\_write\_time 941
  - log\_writes 941
  - sec\_log\_used\_top 1200
  - smallest\_log\_avail\_node 1224
  - tot\_log\_used\_top 1312
  - total\_log\_available 1346
  - total\_log\_used 1347
  - uow\_log\_space\_used 1404
- log writing to disk
  - log\_disk\_wait\_time 936
  - log\_disk\_waits\_total 937
- log\_buffer\_wait\_time 934
- logical data groups 603
- long data
  - long\_object\_pages 942
- long\_object\_l\_pages - Long object data logical pages 942
- machine\_identification 943
- max\_coord\_stmt\_exec\_time 944
- max\_coord\_stmt\_exec\_time\_args 944
- max\_coord\_stmt\_exec\_timestamp 946
- member 958, 1506
- memory usage
  - DB2 pureScale environment 1460
- memory\_free 962
- memory\_pool\_id 962
- memory\_pool\_type 963
- memory\_pool\_used\_hwm 962
- memory\_set\_committed 965
- memory\_set\_id 965
- memory\_set\_size 965
- memory\_set\_type 965
- memory\_set\_used 966
- memory\_set\_used\_hwm 966
- memory\_swap\_free 966
- memory\_swap\_total 967
- memory\_total 967
- messages
  - message 967
- mon\_interval\_id 968
- names
  - db\_name 774, 1492
  - dcs\_db\_name 781
  - service\_subclass\_name 1209
  - service\_superclass\_name 1210
  - work\_action\_set\_name 1425

activities monitor elements (*continued*)

- names (*continued*)
  - work\_class\_name 1426
- network time
  - max\_network\_time\_1\_ms 957
  - max\_network\_time\_100\_ms 956
  - max\_network\_time\_16\_ms 956
  - max\_network\_time\_4\_ms 957
  - max\_network\_time\_500\_ms 958
  - max\_network\_time\_gt500\_ms 958
  - network\_time\_bottom 969
  - network\_time\_top 970
- network\_time\_bottom 969
- network\_time\_top 970
- nicknames
  - create\_nickname 763
  - create\_nickname\_time 764
- no\_change\_updates 971
- nodes
  - coord\_node 750
  - node\_number 971
  - num\_nodes\_in\_db2\_instance 981
  - ss\_node\_number 1239
- nonboundary\_leaf\_node\_splits 972
- num\_db\_storage\_paths 974
- num\_exec\_with\_metrics 975
- num\_extents\_left 975
- num\_extents\_moved 975
- num\_indoubt\_trans 976
- num\_nodes\_in\_db2\_instance 981
- num\_page\_dict\_built 981
- num\_ref\_with\_metrics 981
- num\_references 982
- num\_remaps 982
- num\_tbsps 982
- num\_transmissions 983
- num\_transmissions\_group 984
- numbers
  - progress\_list\_cur\_seq\_num 1161
  - ss\_number 1239
- object\_data\_gbp\_indep\_pages\_found\_in\_lbp 984
- object\_data\_gbp\_invalid\_pages 985
- object\_data\_gbp\_l\_reads 985
- object\_data\_gbp\_p\_reads 986
- object\_data\_l\_reads 987
- object\_data\_lbp\_pages\_found 987
- object\_data\_p\_reads 988
- object\_index\_gbp\_indep\_pages\_found\_in\_lbp 988
- object\_index\_gbp\_invalid\_pages 989
- object\_index\_gbp\_l\_reads 989
- object\_index\_gbp\_p\_reads 990
- object\_index\_l\_reads 991
- object\_index\_lbp\_pages\_found 990
- object\_index\_p\_reads 992
- object\_name 992
- object\_requested 993
- object\_schema 993
- object\_xda\_gbp\_indep\_pages\_found\_in\_lbp 993
- object\_xda\_gbp\_invalid\_pages 994
- object\_xda\_gbp\_l\_reads 995
- object\_xda\_gbp\_p\_reads 995
- object\_xda\_l\_reads 996
- object\_xda\_lbp\_pages\_found 996
- object\_xda\_p\_reads 997
- objects
  - object\_data\_gbp\_invalid\_pages 985
  - object\_name 992

activities monitor elements (*continued*)

- objtype 997, 1510
- OLAP
  - active\_olap\_funcs 644
  - olap\_func\_overflows 998
  - post\_threshold\_olap\_funcs 1147
  - total\_olap\_funcs 1347
- open\_cursors 999
- open\_loc\_curs 999
- open\_loc\_curs\_blk 1000
- open\_rem\_curs 1000
- open\_rem\_curs\_blk 1000
- operations
  - async\_read\_time 678
  - async\_write\_time 678
  - direct\_read\_reqs 791
  - direct\_read\_time 793
  - direct\_reads 795
  - direct\_write\_reqs 797
  - direct\_write\_time 799
  - direct\_writes 801
  - stmt\_operation 1251
- os\_level 1001
- os\_name 1001
- os\_release 1001
- os\_version 1002
- outbound bytes
  - max\_data\_sent\_1024 951
  - max\_data\_sent\_128 952
  - max\_data\_sent\_16384 952
  - max\_data\_sent\_2048 952
  - max\_data\_sent\_256 953
  - max\_data\_sent\_31999 953
  - max\_data\_sent\_4096 954
  - max\_data\_sent\_512 954
  - max\_data\_sent\_64000 955
  - max\_data\_sent\_8192 955
  - max\_data\_sent\_gt64000 955
- outbound bytes received
  - max\_data\_received\_1024 947
  - max\_data\_received\_128 947
  - max\_data\_received\_16384 947
  - max\_data\_received\_2048 948
  - max\_data\_received\_256 948
  - max\_data\_received\_31999 949
  - max\_data\_received\_4096 949
  - max\_data\_received\_512 949
  - max\_data\_received\_64000 950
  - max\_data\_received\_8192 950
  - max\_data\_received\_gt64000 951
- outbound\_bytes\_received 1002
- outbound\_bytes\_received\_bottom 1003
- outbound\_bytes\_received\_top 1003
- outbound bytes sent
  - outbound\_bytes\_sent 1004
  - outbound\_bytes\_sent\_bottom 1004
  - outbound\_bytes\_sent\_top 1004
- outbound communication
  - outbound\_appl\_id 1002
  - outbound\_comm\_address 1004
  - outbound\_comm\_protocol 1005
- outbound sequences
  - outbound\_sequence\_no 1005
- overflow records
  - first\_overflow\_time 850
  - last\_over\_flow\_time 896
  - overflow\_accesses 1005

activities monitor elements (*continued*)

- overflow records (*continued*)
  - overflow\_creates 1006
- overview 521
- overviewtypes 517
- package cache
  - coord\_stmt\_exec\_time 751
  - last\_metrics\_update 896
  - num\_coord\_exec 973
  - num\_coord\_exec\_with\_metrics 973
  - pkg\_cache\_inserts 1021
  - pkg\_cache\_lookups 1022
  - pkg\_cache\_num\_overflow 1024
  - pkg\_cache\_size\_top 1024
  - stmt\_exec\_time 1246
  - stmt\_type\_id 1258
  - total\_routine\_invocations 1356
  - total\_routine\_non\_sect\_proc\_time 1357
  - total\_routine\_non\_sect\_time 1358
  - total\_routine\_time 1359
  - total\_section\_proc\_time 1368
  - total\_section\_time 1374
- package\_elapsed\_time 1006
- package\_id 1006
- package\_list\_count 1007
- package\_list\_exceeded 1007
- package\_list\_size 1007
- packages
  - package\_name 1007
  - package\_schema 1008
  - package\_version\_id 1009
- packet\_receive\_errors 1009
- packet\_send\_errors 1010
- packets\_received 1009
- packets\_sent 1010
- page reclaiming
  - DB2 pureScale environment 1478
- page\_reclaims\_initiated\_s 1012, 1511
- page\_reclaims\_initiated\_x 1012, 1511
- page\_reclaims\_s 1011, 1511
- page\_reclaims\_x 1011, 1512
- pages
  - data\_object\_pages 767
- pages\_read 1014
- pages\_written 1014
- parallelism
  - degree\_parallelism 786
- participant\_no 1016
- participant\_type 1017
- partition\_key 1017
- partitions
  - coord\_partition\_num 751
  - data\_partition\_id 768
  - partition\_number 1018
- pass-through
  - passthru 1018
  - passthru\_time 1018
- past\_activities\_wrapped 1019
- pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbpbuffer
  - pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbp
    - monitor element 1025
- pool\_async\_data\_gbp\_invalid\_pages 1025, 1512
- pool\_async\_data\_gbp\_l\_reads 1026, 1512
- pool\_async\_data\_gbp\_p\_reads 1026, 1513
- pool\_async\_data\_lbp\_pages\_found 1027, 1513

activities monitor elements (*continued*)

- pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbpbuffer pools
  - pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1030
- pool\_async\_index\_gbp\_invalid\_pages 1030, 1514
- pool\_async\_index\_gbp\_l\_reads 1031, 1514
- pool\_async\_index\_gbp\_p\_reads 1031, 1515
- pool\_async\_index\_lbp\_pages\_found 1032, 1515
- pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbpbuffer pools
  - pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1036
- pool\_async\_xda\_gbp\_invalid\_pages 1037, 1516
- pool\_async\_xda\_gbp\_l\_reads 1037, 1516
- pool\_async\_xda\_gbp\_p\_reads 1038, 1517
- pool\_async\_xda\_lbp\_pages\_found 1038, 1517
- pool\_config\_size 1041
- pool\_cur\_size 1042
- pool\_data\_gbp\_indep\_pages\_found\_in\_lbp 1042
- pool\_data\_gbp\_invalid\_pages 1043, 1518
- pool\_data\_gbp\_l\_reads 1045, 1519
- pool\_data\_gbp\_p\_reads 1046, 1521
- pool\_data\_lbp\_pages\_found 1048, 1522
- pool\_failed\_async\_data\_reqs 1058
- pool\_failed\_async\_index\_reqs 1060
- pool\_failed\_async\_other\_reqs 1062
- pool\_failed\_async\_temp\_data\_reqs 1064
- pool\_failed\_async\_temp\_index\_reqs 1066
- pool\_failed\_async\_temp\_xda\_reqs 1068
- pool\_failed\_async\_xda\_reqs 1070
- pool\_id 1072
- pool\_index\_gbp\_indep\_pages\_found\_in\_lbp 1074
- pool\_index\_gbp\_invalid\_pages 1075, 1524
- pool\_index\_gbp\_l\_reads 1076, 1525
- pool\_index\_gbp\_p\_reads 1078, 1527
- pool\_index\_lbp\_pages\_found 1079, 1528
- pool\_queued\_async\_data\_pages 1088
- pool\_queued\_async\_data\_reqs 1090
- pool\_queued\_async\_index\_pages 1092
- pool\_queued\_async\_index\_reqs 1094
- pool\_queued\_async\_other\_reqs 1096
- pool\_queued\_async\_temp\_data\_pages 1098
- pool\_queued\_async\_temp\_data\_reqs 1100
- pool\_queued\_async\_temp\_index\_pages 1102
- pool\_queued\_async\_temp\_index\_reqs 1104
- pool\_queued\_async\_temp\_xda\_pages 1106
- pool\_queued\_async\_temp\_xda\_reqs 1108
- pool\_queued\_async\_xda\_pages 1110
- pool\_queued\_async\_xda\_reqs 1112
- pool\_secondary\_id 1115
- pool\_sync\_data\_gbp\_reads 1116
- pool\_sync\_data\_reads 1116
- pool\_sync\_index\_gbp\_reads 1117
- pool\_sync\_index\_reads 1117
- pool\_sync\_xda\_gbp\_reads 1117
- pool\_sync\_xda\_reads 1117
- pool\_watermark 1128
- pool\_xda\_gbp\_indep\_pages\_found\_in\_lbp 1131
- pool\_xda\_gbp\_invalid\_pages 1132, 1530
- pool\_xda\_gbp\_l\_reads 1133, 1531
- pool\_xda\_gbp\_p\_reads 1135, 1533
- pool\_xda\_lbp\_pages\_found 1139, 1534
- post\_threshold\_peas 1147
- post\_threshold\_peds 1149
- prefetch\_waits 1154

activities monitor elements (*continued*)

- prefetching
  - unread\_prefetch\_pages 1400
- priv\_workspace\_num\_overflows 1157
- progress\_completed\_units 1160
- progress\_work\_metric 1162
- pseudo\_deletes 1163
- pseudo\_empty\_pages 1163
- queries
  - query\_card\_estimate 1164
  - query\_cost\_estimate 1164
  - query\_data\_tag\_list 1165
  - queue\_assignments\_total 1166
  - queue\_size\_top 1166
  - queue\_time\_total 1167
  - select\_time 1204
- query\_actual\_degree 1163
- queue\_start\_time 1166
- queued\_agents 1167
- quiescer
  - quiescer\_auth\_id 1168
  - quiescer\_obj\_id 1168
  - quiescer\_state 1168
  - quiescer\_ts\_id 1169
- ranges
  - bottom 692
  - range\_adjustment 1169
  - range\_container\_id 1169
  - range\_end\_stripe 1169
  - range\_max\_extent 1170
  - range\_max\_page\_number 1170
  - range\_num\_containers 1170
  - range\_number 1170
  - range\_offset 1170
  - range\_start\_stripe 1171
  - range\_stripe\_set\_number 1171
- rebalancing
  - current\_extent 767
- rebinding
  - int\_auto\_rebinds 880
- reclaim\_wait\_time 1171, 1536
- reclaimable\_space\_enabled 1172
- records
  - partial\_record 1015
- reference information 635
- reopt 1177
- reoptimization
  - stmt\_value\_isreopt 1262
- reorg\_completion 1177
- reorg\_long\_tbspc\_id 1178
- reorg\_tbspc\_id 1181
- reorganization
  - page\_reorgs 1010
  - reorg\_current\_counter 1177
  - reorg\_end 1178
  - reorg\_max\_phase 1179
  - reorg\_phase 1179
  - reorg\_phase\_start 1180
  - reorg\_rows\_compressed 1180
  - reorg\_rows\_rejected\_for\_compression 1180
  - reorg\_start 1181
  - reorg\_status 1181
  - reorg\_type 1181
  - reorg\_xml\_regions\_compressed 1182
  - reorg\_xml\_regions\_rejected\_for\_compression 1182
- request\_exec\_time\_avg 1184

activities monitor elements (*continued*)

- requests
  - rqsts\_completed\_total 1198
- response time
  - delete\_time 787
  - host\_response\_time 873
  - insert\_time 879
- roll-forward recovery
  - rf\_log\_num 1184
  - rf\_status 1185
  - rf\_timestamp 1185
  - rf\_type 1185
- rollbacks
  - int\_rollbacks 883
  - rollback\_sql\_stmts 1186
  - rolled\_back\_appl\_id 1187
  - rolled\_back\_participant\_no 1187
  - rolled\_back\_sequence\_no 1187
- routines
  - routine\_id 1188
  - total\_routine\_user\_code\_proc\_time 1360
  - total\_routine\_user\_code\_time 1361
- rows
  - int\_rows\_inserted 885
  - int\_rows\_updated 886
  - rows\_deleted 1189
  - rows\_fetched 1189
  - rows\_inserted 1190
  - rows\_modified 1191
  - rows\_read 1192
  - rows\_returned 1194
  - rows\_selected 1196
  - rows\_updated 1197
  - rows\_written 1197
  - sp\_rows\_selected 1228
- RUNSTATS utility
  - async\_runstats 678
  - sync\_runstats 1266
  - sync\_runstats\_time 1267
- schemas
  - object\_schema 993
- section\_type 1203
- sections
  - priv\_workspace\_section\_inserts 1158
  - priv\_workspace\_section\_lookups 1159
  - section\_actuals 1201
  - section\_env 1202
  - section\_number 1202
  - total\_app\_section\_executions 1318
- sequences
  - progress\_seq\_num 1162
  - sequence\_no 1205
- servers
  - product\_name 1160
  - server\_instance\_name 1206
  - server\_platform 1207
  - server\_prdid 1207
  - server\_version 1208
- service levels
  - service\_level 1209
- service subclasses
  - total\_rqst\_mapped\_in 1363
  - total\_rqst\_mapped\_out 1363
- shared workspaces
  - shr\_workspace\_num\_overflows 1213
  - shr\_workspace\_section\_inserts 1213
  - shr\_workspace\_section\_lookups 1214

activities monitor elements (*continued*)

- shared workspaces (*continued*)
  - shr\_workspace\_size\_top 1214
- skipped\_prefetch\_data\_p\_reads 1215
- skipped\_prefetch\_index\_p\_reads 1216
- skipped\_prefetch\_temp\_data\_p\_reads 1217
- skipped\_prefetch\_temp\_index\_p\_reads 1218
- skipped\_prefetch\_temp\_xda\_p\_reads 1219
- skipped\_prefetch\_uow\_data\_p\_reads 1220
- skipped\_prefetch\_uow\_index\_p\_reads 1220
- skipped\_prefetch\_uow\_temp\_data\_p\_reads 1221
- skipped\_prefetch\_uow\_temp\_index\_p\_reads 1222
- skipped\_prefetch\_uow\_temp\_xda\_p\_reads 1222
- skipped\_prefetch\_uow\_xda\_p\_reads 1222
- skipped\_prefetch\_xda\_p\_reads 1223
- snapshots
  - time\_stamp 1311
- sorting
  - piped\_sorts\_accepted 1020
  - piped\_sorts\_requested 1020
  - post\_shrthreshold\_sorts 1145
  - post\_threshold\_sorts 1151
  - sort\_heap\_allocated 1224
  - sort\_heap\_top 1225
  - sort\_overflows 1225
  - sort\_shrheap\_allocated 1227
  - sort\_shrheap\_top 1227
  - total\_section\_sort\_proc\_time 1369
  - total\_section\_sort\_time 1371
  - total\_section\_sorts 1372
  - total\_sorts 1376, 1385
- source\_service\_class\_id 1228
- spacemappage\_page\_reclaims\_initiated\_s 1236, 1537
- spacemappage\_page\_reclaims\_initiated\_x 1236, 1538
- spacemappage\_page\_reclaims\_s 1235, 1538
- spacemappage\_page\_reclaims\_x 1235, 1539
- spacemappage\_reclaim\_wait\_time 1237, 1540
- SQL communication area (SQLCA)
  - sqlca 1230
- SQL operations
  - elapsed\_exec\_time 808
- SQL statements
  - ddl\_sql\_stmts 782
  - dynamic\_sql\_stmts 805
  - failed\_sql\_stmts 822
  - insert\_sql\_stmts 878
  - num\_compilation 973
  - num\_executions 974
  - select\_sql\_stmts 1204
  - sql\_chains 1229
  - sql\_reqs\_since\_commit 1229
  - sql\_stmts 1230
  - static\_sql\_stmts 1242
  - stmt\_pkgcache\_id 1252
  - stmt\_query\_id 1253
  - stmt\_sorts 1254
  - stmt\_source\_id 1254
  - stmt\_text 1256
  - stmt\_value\_data 1260
  - stmt\_value\_index 1260
  - stmt\_value\_isnull 1261
  - stmt\_value\_type 1262
  - total\_exec\_time 1336
  - uid\_sql\_stmts 1399
- sqlrowsread\_threshold\_id 1231
- sqlrowsread\_threshold\_value 1231
- sqlrowsread\_threshold\_violated 1231

activities monitor elements (*continued*)

- sqlrowsreadinsc\_threshold\_id 1232
- sqlrowsreadinsc\_threshold\_value 1232
- sqlrowsreadinsc\_threshold\_violated 1232
- sqlrowsreturned\_threshold\_id 1233
- sqlrowsreturned\_threshold\_value 1233
- sqlrowsreturned\_threshold\_violated 1233
- sqltempespace\_threshold\_value 1234
- sqltempespace\_threshold\_violated 1234
- statements
  - prep\_time\_best 1156
  - prep\_time\_worst 1156
  - stmt\_first\_use\_time 1247
  - stmt\_history\_id 1247
  - stmt\_history\_list\_size 1248
  - stmt\_invocation\_id 887, 1248
  - stmt\_isolation 1249
  - stmt\_last\_use\_time 1249
  - stmt\_nest\_level 968, 1250
  - stmt\_node\_number 1251
  - stmt\_type 1257
- status
  - db\_status 776
  - db2\_status 772
  - dcs\_appl\_status 781
  - ss\_status 1239
- stmt\_unicode 1259
- storage paths
  - num\_db\_storage\_paths 974
- storage\_group\_id 1264
- storage\_group\_name 1264
- stored procedures
  - stored\_proc\_time 1264
  - stored\_procs 1265
- stripe sets
  - container\_stripe\_set 740
- swap\_page\_size 1266
- swap\_pages\_in 1265
- swap\_pages\_out 1265
- system\_auth\_id 1268
- table queues
  - tq\_tot\_send\_spills 1397
- table spaces
  - index\_tbsp\_id 878
  - long\_tbsp\_id 943
  - rebalancer\_extents\_processed 1285
  - rebalancer\_extents\_remaining 1285
  - rebalancer\_last\_extent\_moved 1286
  - rebalancer\_mode 1286
  - rebalancer\_priority 1287
  - rebalancer\_restart\_time 1288
  - rebalancer\_start\_time 1289
  - rebalancer\_status 1289
  - rebalancer\_target\_storage
    - \_group\_id 1290
  - rebalancer\_target\_storage
    - \_group\_name 1290
  - tablespace\_auto\_resize\_enabled 1274
  - tablespace\_content\_type 1275
  - tablespace\_cur\_pool\_id 1275
  - tablespace\_current\_size 1276
  - tablespace\_extent\_size 1276
  - tablespace\_free\_pages 1276
  - tablespace\_id 1277
  - tablespace\_increase\_size 1278
  - tablespace\_increase\_size\_percent 1278
  - tablespace\_initial\_size 1278

activities monitor elements (*continued*)

table spaces (*continued*)

- tablespace\_last\_resize\_failed 1279
- tablespace\_last\_resize\_time 1279
- tablespace\_max\_size 1279
- tablespace\_min\_recovery\_time 1280, 1544
- tablespace\_name 1280
- tablespace\_next\_pool\_id 1281
- tablespace\_num\_containers 1282
- tablespace\_num\_quiescers 1282
- tablespace\_num\_ranges 1282
- tablespace\_page\_size 1282
- tablespace\_page\_top 1283
- tablespace\_pending\_free\_pages 1284
- tablespace\_prefetch\_size 1284
- tablespace\_rebalancer\_extents\_processed 1285
- tablespace\_rebalancer\_extents\_remaining 1285
- tablespace\_rebalancer\_last\_extent\_moved 1286
- tablespace\_rebalancer\_mode 1286
- tablespace\_rebalancer\_priority 1287
- tablespace\_rebalancer\_restart\_time 1288
- tablespace\_rebalancer\_source\_storage
  - \_group\_id 1288
- tablespace\_rebalancer\_source\_storage
  - \_group\_name 1288
- tablespace\_rebalancer\_start\_time 1289
- tablespace\_rebalancer\_status 1289
- tablespace\_rebalancer\_target\_storage
  - \_group\_id 1290
- tablespace\_rebalancer\_target\_storage
  - \_group\_name 1290
- tablespace\_state 1291
- tablespace\_state\_change\_object\_id 1293
- tablespace\_state\_change\_ts\_id 1293
- tablespace\_total\_pages 1293
- tablespace\_type 1294
- tablespace\_usable\_pages 1294
- tablespace\_used\_pages 1295
- tablespace\_using\_auto\_storage 1295
- tbsp\_auto\_resize\_enabled 1274
- tbsp\_content\_type 1275
- tbsp\_cur\_pool\_id 1275
- tbsp\_current\_size 1276
- tbsp\_datatag 1297
- tbsp\_extent\_size 1276
- tbsp\_free\_pages 1276
- tbsp\_id 1277
- tbsp\_increase\_size 1278
- tbsp\_increase\_size\_percent 1278
- tbsp\_initial\_size 1278
- tbsp\_last\_resize\_failed 1279
- tbsp\_last\_resize\_time 1279
- tbsp\_max\_page\_top 1297
- tbsp\_max\_size 1279
- tbsp\_min\_recovery\_time 1280, 1544
- tbsp\_next\_pool\_id 1281
- tbsp\_num\_containers 1282
- tbsp\_num\_quiescers 1282
- tbsp\_num\_ranges 1282
- tbsp\_page\_size 1282
- tbsp\_page\_top 1283
- tbsp\_pending\_free\_pages 1284
- tbsp\_prefetch\_size 1284
- tbsp\_rebalancer\_extents\_processed 1285
- tbsp\_rebalancer\_extents\_remaining 1285
- tbsp\_rebalancer\_last\_extent\_moved 1286
- tbsp\_rebalancer\_mode 1286

activities monitor elements (*continued*)

- table spaces (*continued*)
  - tbsp\_rebalancer\_priority 1287
  - tbsp\_rebalancer\_restart\_time 1288
  - tbsp\_rebalancer\_start\_time 1289
  - tbsp\_rebalancer\_status 1289
  - tbsp\_rebalancer\_target\_storage\_group\_id 1290
  - tbsp\_rebalancer\_target\_storage\_group\_name 1290
  - tbsp\_state 1291
  - tbsp\_state\_change\_object\_id 1293
  - tbsp\_state\_change\_ts\_id 1293
  - tbsp\_total\_pages 1293
  - tbsp\_trackmod\_state 1298
  - tbsp\_type 1294
  - tbsp\_usable\_pages 1294
  - tbsp\_used\_pages 1295
  - tbsp\_using\_auto\_storage 1295
  - ts\_name 1398
- tables
  - tab\_file\_id 1269
  - tab\_type 1269
  - table\_file\_id 1269
  - table\_name 1270, 1541
  - table\_scans 1272
  - table\_schema 1272, 1542
  - table\_type 1274
- tablespace\_paths\_dropped 1283
- target\_cf\_gbp\_size 1296, 1545
- target\_cf\_lock\_size 1296, 1545
- target\_cf\_sca\_size 1296, 1545
- tbsp\_last\_consec\_page 1297
- TCP/IP
  - tcPIP\_sends\_total 1303
- tcPIP\_send\_volume 1301
- tcPIP\_send\_wait\_time 1302
- territory\_code 1305
- thresholds
  - num\_lw\_thresh\_exceeded 980
  - num\_threshold\_violations 983
  - thresh\_violations 1305
  - threshold\_action 1307
  - threshold\_domain 1307
  - threshold\_maxvalue 1308
  - threshold\_name 1308
  - threshold\_predicate 1308
  - threshold\_queuesize 1309
  - thresholdid 1310
- time
  - evmon\_wait\_time 815
  - prefetch\_wait\_time 1152
  - prep\_time 1155
  - progress\_start\_time 1162
  - ss\_exec\_time 1238
  - stmt\_elapsed\_time 1246
  - time\_completed 1310
  - time\_created 1311
  - time\_of\_violation 1311
  - time\_started 1311
  - total\_sort\_time 1375
- time spent
  - examples of usage 539
  - overview 529
  - ranking 539
  - viewing time spent during SQL statement execution 543

activities monitor elements (*continued*)

- time spent (*continued*)
  - viewing time spent in activities 543
  - viewing time spent in the system 539
- time stamps
  - activate\_timestamp 644
  - db\_conn\_time 773
  - db2start\_time 773
  - last\_backup 895
  - last\_reset 897
  - lock\_wait\_start\_time 924
  - message\_time 968
  - statistics\_timestamp 1242
  - status\_change\_time 1245
  - stmt\_start 1255
  - stmt\_stop 1255
- time zones
  - time\_zone\_disp 1312
- tokens
  - consistency\_token 738
  - corr\_token 752
- total\_app\_commits 1316
- total\_app\_rollbacks 1316
- total\_bytes\_received 1320
- total\_bytes\_sent 1321
- total\_commit\_proc\_time 1321
- total\_commit\_time 1322
- total\_compilations 1323
- total\_compile\_proc\_time 1324
- total\_compile\_time 1325
- total\_connect\_authentication\_proc\_time 1326
- total\_connect\_authentication\_time 1328
- total\_connect\_authentications 1327
- total\_connect\_request\_proc\_time 1329
- total\_connect\_request\_time 1331
- total\_connect\_requests - Connection or switch user requests 1330
- total\_extended\_latch\_wait\_time 1336
- total\_extended\_latch\_waits 1338
- total\_hash\_loops 1340
- total\_implicit\_compilations 1340
- total\_implicit\_compile\_proc\_time 1341
- total\_implicit\_compile\_time 1342
- total\_load\_proc\_time 1343
- total\_load\_time 1344
- total\_loads 1345
- total\_move\_time 1339, 1347
- total\_peas 1348
- total\_peds 1350
- total\_reorg\_proc\_time 1351
- total\_reorg\_time 1352
- total\_reorgs 1353
- total\_rollback\_proc\_time 1354
- total\_rollback\_time 1355
- total\_runstats 1365
- total\_runstats\_proc\_time 1366
- total\_runstats\_time 1367
- total\_stats\_fabrication\_proc\_time 1378
- total\_stats\_fabrication\_time 1379
- total\_stats\_fabrications 1380
- total\_sync\_runstats 1383
- total\_sync\_runstats\_proc\_time 1382
- total\_sync\_runstats\_time 1381
- tq\_cur\_send\_spills 1391
- tq\_id\_waiting\_on 1391
- tq\_max\_send\_spills 1391
- tq\_node\_waited\_for 1392

activities monitor elements (*continued*)

- tq\_rows\_read 1392
- tq\_rows\_written 1393
- tq\_sort\_heap\_rejections 1393
- tq\_sort\_heap\_requests 1395
- tq\_wait\_for\_any 1398
- transactions
  - client\_acctng 709
  - client\_userid 717
  - client\_wrkstnname 718
  - num\_indoubt\_trans 976
  - tpmon\_acc\_str 1388
  - tpmon\_client\_userid 1390
  - tpmon\_client\_wrkstn 1390
  - xid monitor 1432
- units of work (UOW)
  - completion\_status 722
  - parent\_uow\_id 1015
  - prev\_uow\_stop\_time 1156
  - progress\_total\_units 1162
  - uow\_comp\_status 1400
  - uow\_completed\_total 1401
  - uow\_elapsed\_time 1402
  - uow\_id 1402
  - uow\_lifetime\_avg 1403
  - uow\_start\_time 1405
  - uow\_status 1406
  - uow\_stop\_time 1406
  - uow\_throughput 1407
- updates
  - update\_sql\_stmts 1408
- usage lists
  - usage\_list\_last\_state\_change 1409
  - usage\_list\_last\_updated 1409
  - usage\_list\_mem\_size 1409
  - usage\_list\_name 1410
  - usage\_list\_num\_ref\_with\_metrics 1410
  - usage\_list\_num\_references 1410
  - usage\_list\_schema 1411
  - usage\_list\_size 1411
  - usage\_list\_state 1411
  - usage\_list\_used\_entries 1411
  - usage\_list\_wrapped 1412
- usage\_list\_last\_state\_change 1409
- usage\_list\_last\_updated 1409
- usage\_list\_mem\_size 1409
- usage\_list\_name 1410
- usage\_list\_num\_ref\_with\_metrics 1410
- usage\_list\_num\_references 1410
- usage\_list\_schema 1411
- usage\_list\_size 1411
- usage\_list\_state 1411
- usage\_list\_used\_entries 1411
- usage\_list\_wrapped 1412
- utilities
  - utility\_dbname 1412
  - utility\_description 1413
  - utility\_id 1413
  - utility\_invoker\_type 1414
  - utility\_priority 1417
  - utility\_start\_time 1417
  - utility\_state 1417
  - utility\_type 1418
- valid 1418, 1419
- virtual\_mem\_free 1420
- virtual\_mem\_reserved 1421
- virtual\_mem\_total 1421

activities monitor elements (*continued*)

- wait
  - evmon\_waits\_total 817
- wait times
  - diaglog\_write\_wait\_time 789
  - lock\_wait\_time\_top 929
  - prefetch\_wait\_time 1152
  - total\_wait\_time 1387
- wait timescomponent processing time spentcomponent
  - elapsed time spenttime spent
  - hierarchy 531
- watermarks
  - act\_cpu\_time\_top 639
  - act\_rows\_read\_top 641
  - concurrent\_act\_top 725
  - concurrent\_connection\_top 726
  - concurrent\_wlo\_act\_top 726
  - concurrent\_wlo\_top 727
  - coord\_act\_lifetime\_top 746
  - cost\_estimate\_top 752
  - lock\_wait\_time\_top 929
  - rows\_returned\_top 1195
  - temp\_tablespace\_top 1304
  - uow\_total\_time\_top 1407
- WLM dispatcher
  - cpu\_limit 756
  - cpu\_share\_type 757
  - cpu\_shares 757
  - cpu\_utilization 760
  - cpu\_velocity 760
  - estimated\_cpu\_entitlement 809
  - total\_disp\_run\_queue\_time 1334
- workload management
  - total queue assignments 1422
  - total queue time 1423
  - wl\_work\_action\_set\_id 1421
  - wl\_work\_class\_id 1422
- workloads
  - wlo\_completed\_total 1425
  - workload\_id 1427
  - workload\_name 1428
  - workload\_occurrence\_id 1429
  - workload\_occurrence\_state 1429
- xda\_object\_l\_pages - XML storage object (XDA) data logical
  - pages 1431
- xmlid 1432
- XQuery
  - xquery\_stmts 1432
- activity event monitors
  - accessing data written to tables 269
  - configuring data collection 256
  - creating 255
  - data captured 258
  - data returned
    - table event monitors 258
  - monitor data returned in XML documents 13
  - overview 253
- activity metrics
  - activities event monitor
    - data captured 258
  - See activity monitor elements 521
- activity throughput
  - monitor elements
    - act\_throughput 643
- ACTIVITYTOTALTIME activity threshold
  - monitor elements
    - activitytotaltime\_threshold\_id 648

ACTIVITYTOTALTIME activity threshold (*continued*)  
 monitor elements (*continued*)  
   activitytotaltime\_threshold\_value 648  
   activitytotaltime\_threshold\_violated 649

agents  
 monitor elements  
   agent\_id 650  
   agent\_id\_holding\_lock 651  
   agent\_pid 652  
   agent\_status 652  
   agent\_sys\_cpu\_time 653  
   agent\_usr\_cpu\_time 654  
   agent\_wait\_time 654  
   agent\_waits\_total 655  
   agents\_created\_empty\_pool 656  
   agents\_from\_pool 657  
   agents\_registered 657  
   agents\_registered\_top 658  
   agents\_stolen 658  
   agents\_top 658  
   agents\_waiting\_on\_token 659  
   agents\_waiting\_top 659  
   appl\_priority 671  
   associated\_agents\_top 678  
   coord\_agent\_pid 749  
   coord\_agents\_top 749  
   idle\_agents 873  
   locks\_waiting 934  
   max\_agent\_overflows 943  
   num\_agents 972  
   num\_assoc\_agents 972  
   priv\_workspace\_size\_top 1159  
   quiescer\_agent\_id 1167  
   rolled\_back\_agent\_id 1186

alert actions  
 health indicators  
   states 509

alerts  
 DB2 pureScale environments  
   hosts 1447  
   interpreting 1441  
   values 1439  
   viewing details 1448, 1454  
 enabling 488  
 resolving  
   GET RECOMMENDATIONS command 502  
   SQL queries 499  
 retrieving recommendations  
   client application 499

aliases  
 input\_db\_alias element monitor element 878

ALTER EVENT MONITOR statement  
 example 128

API request types  
 health monitor 487  
 snapshot monitor 425

appl\_status monitor element 673

applications  
 monitor elements  
   appl\_id 666  
   appl\_id\_holding\_lk 668  
   appl\_id\_oldest\_xact 669  
   appl\_idle\_time 670  
   appl\_name 670  
   appl\_priority 671  
   appl\_priority\_type 672  
   appl\_section\_inserts 672

applications (*continued*)  
 monitor elements (*continued*)  
   appl\_section\_lookups 673  
   appl\_status 673  
   application\_handle 675  
   appls\_cur\_cons 677  
   appls\_in\_db2 677  
   client\_applname 710  
   creator 764  
   rolled\_back\_participant\_no 1187  
   tpmon\_client\_app 1389

async\_read\_time monitor element 678  
 async\_write\_time monitor element 678

attributes  
 progress\_list\_attr monitor element 1161

audits  
 monitor elements  
   audit\_events\_total 679  
   audit\_file\_write\_wait\_time 680  
   audit\_file\_writes\_total 682

authorization IDs  
 monitor elements  
   auth\_id 686  
   execution\_id 821  
   quiescer\_auth\_id 1168  
   session\_auth\_id 1212

authorization level monitor element 688

automatic storage paths  
 monitor elements  
   db\_storage\_path 776  
   sto\_path\_free\_sz 1263

## B

backups  
 databases  
   Database Backup Required health indicator 474  
   db.db\_backup\_req health indicator 474  
   last\_backup monitor element 895

buffer pool event monitor  
 data returned  
   logical data groups 350  
   table event monitors 350

buffer pool hit ratios 1469

buffer pools  
 DB2 pureScale  
   monitoring overview 1465  
 DB2 pureScale environments  
   calculating hit ratios 1472  
   hit ratio overview 1467  
   monitoring 1465  
   temporary buffer pools 1472

hit ratios 449

monitor elements 449  
 automatic 689  
 block\_ios 690  
 bp\_cur\_buffsz 693  
 bp\_id 693  
 bp\_name 693  
 bp\_new\_buffsz 694  
 bp\_pages\_left\_to\_remove 694  
 bp\_tbsp\_use\_count 694  
 buff\_free 695  
 buff\_free\_bottom 695  
 object\_data\_l\_reads 987  
 object\_data\_p\_reads 988  
 object\_index\_l\_reads 991

- buffer pools (*continued*)
  - monitor elements (*continued*)
    - object\_index\_p\_reads 992
    - object\_xda\_l\_reads 996
    - object\_xda\_p\_reads 997
    - pool\_async\_data\_read\_reqs 1027
    - pool\_async\_data\_reads 1028
    - pool\_async\_data\_writes 1029
    - pool\_async\_index\_read\_reqs 1032
    - pool\_async\_index\_reads 1033
    - pool\_async\_index\_writes 1034
    - pool\_async\_read\_time 1034
    - pool\_async\_write\_time 1035
    - pool\_async\_xda\_gbp\_invalid\_pages 1037, 1516
    - pool\_async\_xda\_gbp\_l\_reads 1037, 1516
    - pool\_async\_xda\_gbp\_p\_reads 1038, 1517
    - pool\_async\_xda\_lbp\_pages\_found 1038, 1517
    - pool\_async\_xda\_read\_reqs 1039
    - pool\_async\_xda\_reads 1039
    - pool\_async\_xda\_writes 1040
    - pool\_data\_l\_reads 1049
    - pool\_data\_p\_reads 1051
    - pool\_data\_writes 1053
    - pool\_drty\_pg\_steal\_clns 1056
    - pool\_drty\_pg\_thrsh\_clns 1057
    - pool\_index\_l\_reads 1081
    - pool\_index\_p\_reads 1083
    - pool\_index\_writes 1085
    - pool\_lsn\_gap\_clns 1087
    - pool\_no\_victim\_buffer 1088
    - pool\_read\_time 1114
    - pool\_temp\_data\_l\_reads 1117
    - pool\_temp\_data\_p\_reads 1119
    - pool\_temp\_index\_l\_reads 1121
    - pool\_temp\_index\_p\_reads 1122
    - pool\_temp\_xda\_l\_reads 1124
    - pool\_temp\_xda\_p\_reads 1126
    - pool\_write\_time 1129
    - pool\_xda\_gbp\_invalid\_pages 1132, 1530
    - pool\_xda\_gbp\_l\_reads 1133, 1531
    - pool\_xda\_gbp\_p\_reads 1135, 1533
    - pool\_xda\_l\_reads 1137
    - pool\_xda\_lbp\_pages\_found 1139, 1534
    - pool\_xda\_p\_reads 1140
    - pool\_xda\_writes 1142
    - tablespace\_cur\_pool\_id 1275
    - tablespace\_next\_pool\_id 1281
    - tbsp\_cur\_pool\_id 1275
    - tbsp\_next\_pool\_id 1281
  - monitoring
    - DB2 pureScale environments 1465
  - pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1025
  - pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1030
  - pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1036
  - temporary in DB2 pureScale instances 1472
- buffers
  - num\_log\_data\_found\_in\_buffer monitor elements 978
- built-in views
  - DB2\_CF
    - overview 1437
  - DB2\_CLUSTER\_HOST\_STATE
    - overview 1437
  - DB2\_INSTANCE\_ALERTS
    - overview 1437

- built-in views (*continued*)
  - DB2\_MEMBER
    - overview 1437
- byte order
  - byte\_order monitor element 697

## C

- caching
  - stats\_cache\_size monitor element 1243
- catalog cache
  - db.catcache\_hitratio health indicator 480
  - monitor elements
    - cat\_cache\_inserts 698
    - cat\_cache\_lookups 699
    - cat\_cache\_overflows 700
    - cat\_cache\_size\_top 701
- catalog nodes
  - monitor elements
    - catalog\_node 702
    - catalog\_node\_name 702
- ch\_free monitor element 707
- change history
  - monitor elements
    - backup\_timestamp 689
    - cfg\_collection\_type 704
    - cfg\_name 704
    - cfg\_old\_value 705
    - cfg\_old\_value\_flags 705
    - cfg\_value 706
    - cfg\_value\_flags 706
    - ddl\_classification 781
    - deferred 786
    - device\_type 788
    - location 901
    - location\_type 902
    - phase\_start\_event\_id 1019
    - phase\_start\_event\_timestamp 1019
    - regvar\_collection\_type 1173
    - regvar\_level 1173
    - regvar\_name 1173
    - regvar\_old\_value 1174
    - regvar\_value 1174
    - savepoint\_id 1199
    - start\_event\_id 1241
    - start\_event\_timestamp 1241
    - tbsp\_names 1298
    - txn\_completion\_status 1399
    - utility\_detail 1413
    - utility\_invocation\_id 1414
    - utility\_operation\_type 1415
    - utility\_phase\_detail 1416
    - utility\_phase\_type 1416
    - utility\_start\_type 1417
    - utility\_stop\_type 1418
- change history event monitor
  - CHANGESUMMARY\_evmon-name 372
  - data returned
    - logical data groups 367
  - DBDBMCFG\_evmon-name 375
  - DDLSTMTEXEC\_evmon-name 378
  - EVMONSTART\_evmon-name 382
  - listing changes performed by the STMM example 402
  - listing committed DDL statements example 401
  - logical data groups
    - CHANGESUMMARY 372
    - DBDBMCFG 375

- change history event monitor (*continued*)
  - logical data groups (*continued*)
    - DDLSTMTEXEC 378
    - EVMONSTART 382
    - REGVAR 377
    - TXNCOMPLETION 381
    - UTILLOCATION 387
    - UTILPHASE 391
    - UTILSTART 383
    - UTILSTOP 389
  - monitoring changes performed by the STMM
    - example 402
  - monitoring committed DDL statements example 401
  - monitoring configuration changes example 398
  - monitoring LOAD operations example 399
  - monitoring utility execution example 400
  - monitoring utility executions example 398
  - REGVAR\_evmon-name 377
  - TXNCOMPLETION\_evmon-name 381
  - usage example 396
  - UTILLOCATION\_evmon-name 387
  - UTILSTART\_evmon-name 383, 391
  - UTILSTOP\_evmon-name 389
- CHANGESUMMARY
  - change history event monitor
    - logical data groups 372
- client applications
  - health snapshots 493
- client operating platform monitor element 714
- client process ID monitor element 713
- client product and version ID monitor element 715
- cluster caching facilities
  - alerts
    - interpreting 1441
    - values 1439
  - memory
    - monitor elements 1460
    - monitoring usage 1459
    - viewing usage 1461
  - monitor elements
    - memory 1460
  - monitoring
    - memory usageCPU load 1459
  - processor loads 1464
  - states
    - interpreting 1441
    - values 1439
  - status
    - viewing 1448
- code pages
  - monitor elements
    - codepage\_id 718
    - host\_ccsid 870
- coded character set identifier (CCSID)
  - host\_ccsid monitor element 870
- collection levels
  - monitor elements 525
- command line processor (CLP)
  - commands
    - health monitor 486
    - health snapshot capturing 493
- commit statements attempted monitor element 721
- commits
  - int\_commits monitor element 881
- communication error time monitor element
  - gw\_comm\_error\_time element 853
- communication errors monitor element
  - gw\_comm\_errors element 853
- communication protocols
  - client\_protocol monitor element 716
- completed progress work units monitor element 1160
- component elapsed times
  - monitor elements 531
  - viewing
    - activity-level examples 543
    - system-level examples 539
- component processing times
  - monitor elements 531
  - viewing
    - activity-level examples 543
    - system-level examples 539
- con\_response\_time monitor element 725
- configuration
  - .db2toprc file 436
- configuration parameters
  - monitor element collection levels 525
- connections
  - monitor elements
    - appl\_con\_time 666
    - appls\_cur\_cons 677
    - appls\_in\_db2 677
    - con\_elapsed\_time 724
    - con\_local\_databases 724
    - conn\_complete\_time 736
    - conn\_time 736
    - connection\_status 737
    - connections\_top 738
    - dl\_conns 805
    - gw\_connections\_top 854
    - gw\_cons\_wait\_client 854
    - gw\_cons\_wait\_host 854
    - gw\_cur\_cons 855
    - gw\_total\_cons 856
    - local\_cons 900
    - local\_cons\_in\_exec 900
    - num\_gw\_conn\_switches 975
    - rem\_cons\_in 1175
    - rem\_cons\_in\_exec 1175
    - total\_sec\_cons 1367
- connections event monitor
  - data returned
    - logical data groups 354
    - table event monitors 355
- containers
  - monitor elements
    - container\_accessible 739
    - container\_id 739
    - container\_name 739
    - container\_total\_pages 740
    - container\_type 741
    - container\_usable\_pages 741
- control tables
  - event monitors 89
- counters
  - data element type 445
- cpu limit
  - monitor elements
    - cpu\_limit 756
- cpu share type
  - monitor elements
    - cpu\_share\_type 757

- cpu shares
  - monitor elements
    - cpu\_shares 757
- CPU time
  - monitor elements
    - agent\_sys\_cpu\_time 653
    - agent\_usr\_cpu\_time 654
    - ss\_sys\_cpu\_time 1240
    - ss\_usr\_cpu\_time 1240
    - stmt\_sys\_cpu\_time 1256
    - stmt\_usr\_cpu\_time 1259
    - system\_cpu\_time 1268
    - total\_cpu\_time 1332
    - total\_sys\_cpu\_time 1385
    - total\_usr\_cpu\_time 1387
    - user\_cpu\_time 1412
- cpu utilization
  - monitor elements
    - cpu\_utilization 760
- CPU utilization
  - monitor elements
    - cpu\_idle 754
    - cpu\_iowait 755
    - cpu\_system 757
    - cpu\_usage\_total 759
    - cpu\_user 759
- cpu velocity
  - monitor elements
    - cpu\_velocity 760
- CPUs
  - See also* processors
  - cluster caching facilities
    - load monitoring 1459
- creating event monitors
  - table 37
  - unformatted event table 96
- creator monitor element 764
- cursors
  - monitor elements
    - acc\_curs\_blk 636
    - blocking\_cursor 691
    - cursor\_name 767
    - open\_cursors 999
    - open\_loc\_curs 999
    - open\_loc\_curs\_blk 1000
    - open\_rem\_curs 1000
    - open\_rem\_curs\_blk 1000
    - rej\_curs\_blk 1174
- customizing
  - MONREPORT module 408
  - reports
    - MONREPORT module 408

**D**

- data objects
  - monitoring 523
- data partitions
  - data\_partition\_id monitor element 768
- data representation
  - element types
    - counters 445
    - overview 444
- data sources
  - data source name monitor element 769
  - health indicator 483

- database event monitor
  - data returned
    - logical data groups 334
    - table event monitors 334
- database objects
  - getting usage statistics 10
  - monitoring
    - object usage 8
    - objects that a statement affects 10
    - statements that affect a table 8
  - usage 8
- database paths
  - db\_path element monitor element 775
- database system monitor
  - data organization 444
  - information restricting 438
  - interfaces 451
  - memory requirements 447
  - output 446
  - sample 451
  - self-describing data stream 446
- database-managed space (DMS)
  - table spaces
    - health indicators 462
- databases
  - aliases
    - application monitor element 711
    - gateway monitor element 855
  - connections
    - connects since database activation monitor element 1326
  - local
    - con\_local\_dbases monitor element 724
  - monitor elements
    - application 711
    - connects since database activation 1326
    - database deactivation timestamp 804
    - gateway 855
    - monitoring 1
      - overview 3
- datasource\_name element 769
- db\_heap\_top monitor element 774
- db\_status monitor element 776
- db.lock\_escal\_rate health indicator 479
- db.locklist\_utilization health indicator 478
- DB2 Connect
  - monitor elements
    - gw\_con\_time 853
    - gw\_cur\_cons 855
    - gw\_exec\_time 855
    - gw\_total\_cons 856
- DB2 Information Center
  - updating 1552, 1554
  - versions 1552
- DB2 Performance Counters 513
- DB2 pureScale
  - alerts
    - hosts 1447
    - interpreting 1441
    - values 1439
    - viewing details 1454
  - buffer pools
    - calculating hit ratios 1472
    - hit rates 1467
    - hit ratios 1467
    - monitoring 1465
  - event monitoring 114

DB2 pureScale (continued)

- locks
  - between members 1475
  - lock waits 1475
  - monitoring 1475
  - overview 1475
- monitoring
  - buffer pool hit rates 1467
  - buffer pool hit ratios 1467
  - buffer pools 1465
  - databases 1457
  - events 1457
  - locks 1475
  - overview 1435, 1437
  - systems 1457
- server status 1435
- states
  - interpreting 1441
  - values 1439
- DB2 pureScale instances
  - cluster caching facility status 1448
  - hosts
    - status 1447
  - member status 1452
  - members
    - status 1448
  - monitoring status 1437
  - status
    - cluster caching facilities 1448
    - hosts 1447
    - members 1448, 1452
    - monitoring 1437
    - overview 1447
    - retrieval interfaces 1437
  - troubleshooting
    - status monitoring 1437
- DB2 workload management
  - monitor elements
    - total queue assignments 1422
    - total queue time 1423
- DB2\_CF administrative view
  - overview 1437
- DB2\_CLUSTER\_HOST\_STATE administrative view
  - overview 1437
- DB2\_GET\_CLUSTER\_HOST\_STATE table function
  - overview 1437
- DB2\_GET\_INSTANCE\_INFO table function
  - overview 1437
- DB2\_INSTANCE\_ALERTS administrative view
  - overview 1437
  - viewing alert details 1454
- DB2\_MEMBER administrative view
  - overview 1437
- db2\_status monitor element 772
- db2advis command
  - input file
    - creating with the package cache event monitor 248
- db2cluster command
  - retrieving DB2 pureScale instance status 1437
  - viewing alerts 1448, 1454
- DB2DETAILDEADLOCK event monitor
  - disabling 129
- db2event.ctl control file 103
- db2evmon command
  - handling large data streams 106
- db2evmonfmt tool 243
  - details 120
- db2evmonfmt tool (continued)
  - lock event data 160
  - unit of work event data 209
- db2instance command
  - example 1454
  - retrieving DB2 pureScale instance status 1437
  - viewing DB2 pureScale instance status 1448
- db2perf command
  - resetting database performance values 515
- db2perfi command
  - installing and registering DB2Perf.DLL 513
- db2perfr command
  - registering administrator user name and password with DB2 513
- db2top command
  - monitoring 433
- DBDBMCFG
  - change history event monitor
    - logical data groups 375
- dcs\_app1\_status monitor element 781
- DDLSTMTEXEC
  - change history event monitor
    - logical data groups 378
- deadlock event monitor
  - data returned
    - table event monitors 363
- deadlocks
  - db.deadlock\_rate health indicator 477
  - deprecated features
    - DB2 pureScale environment 1483
  - monitor elements
    - deadlock\_id 783
    - deadlock\_node 784
    - deadlocks 784
    - dl\_conns 805
    - int\_deadlock\_rollbacks 882
    - participant\_no 1016
  - reports 160
- DELETE statement
  - delete\_sql\_stmts monitor element 787
- deprecated functionality
  - deadlock event monitor
    - DB2 pureScale environment 1483
  - LIST TABLESPACE CONTAINERS command
    - DB2 pureScale environment 1483
  - LIST TABLESPACES command
    - DB2 pureScale environment 1483
  - monitoring tools 455
  - snapshot monitor
    - DB2 pureScale environment 1483
- descriptors
  - progress\_description monitor element 1161
- Design Advisor
  - creating input file with the package cache event monitor 248
- DETAILS.XML
  - monitor table functions 13
- disconn\_time element 804
- documentation
  - overview 1549
  - PDF files 1549
  - printed 1549
  - terms and conditions of use 1556

## E

- ENV\_CF\_SYS\_RESOURCES administrative view
  - example
    - viewing cluster caching facility processor load 1464
- environment handles
  - comp\_env\_desc monitor element 722
- errors
  - gw\_comm\_errors monitor element 853
- estimated cpu entitlement
  - monitor elements
    - estimated\_cpu\_entitlement 809
- event monitors
  - accessing data
    - regular tables 119
  - activities
    - data written to tables 258
  - activities event monitors
    - configuring data collection 256
  - activity
    - creating 255
    - overview 253
  - activity event monitors
    - accessing data written to tables 269
- blocked
  - overview 107
- buffer pool
  - data written to tables 350
  - logical data groups 350
- buffers 107
- change history
  - logical data groups 367
  - overview 366
  - usage examples 395
- changing 128
- comparison of UE and regular table output 100
- configuring data collection
  - activities event monitors 256
- connections
  - data written to tables 355
  - logical data groups 354
- control tables 89
- creating
  - activity event monitors 255
  - file event monitors 101
  - named pipe event monitors 105
  - overview 33
- creating in DB2 pureScale environment
  - for partitioned databases 114
- data written to tables
  - activities event monitor 258
  - buffer pool event monitor 350
  - connections event monitor 355
  - database event monitor 334
  - deadlock event monitor 363
  - locking event monitor 132
  - package cache event monitor 219
  - statement event monitor 344
  - statistics event monitor 274
  - table event monitor 348
  - table space event monitor 352
  - threshold violations event monitor 342
  - transaction event monitor 360
  - unit of work event monitor 166
- database
  - data written to tables 334
  - logical data groups 334
- db2evmonfmt Java-based tool for parsing data 120
- event monitors (*continued*)
  - deadlock
    - data written to tables 363
    - logical data groups 362
  - elements
    - count 753
    - event\_monitor\_name 812
    - evmon\_activates 814
    - evmon\_flushes 820
  - event type to logical data group mappings 110, 596
  - event\_type monitor element event types 813
  - events captured 27
  - executable listing 206
  - file management 103
  - implications of not upgrading 403
  - listing 113
  - locking
    - data written to tables 132
    - logical data groups 132
    - overview 129
    - usage example 160
  - logical data groups 40, 547
    - buffer pool event monitor 350
    - change history event monitor 367
    - changing 128
    - connections event monitor 354
    - database event monitor 334
    - locking event monitor 132
    - package cache event monitor 218
    - statement event monitor 343
    - table event monitor 348
    - table space event monitor 352
    - threshold violation event monitor 342
    - unit of work event monitor 166
  - monitor element list 40, 547
  - named pipe management 106
  - non-blocked
    - overflow records 89
    - overview 107
  - output
    - pruning 126
    - self-describing data stream 108
  - output options
    - details 34
  - overflow records 89
  - overview 27
  - package cache
    - data written to tables 219
    - logical data groups 218
    - overview 216
  - package listing
    - unit of work event monitor 201
  - statement
    - data written to tables 344
    - logical data groups 343
  - statistics
    - data written to tables 274
    - overview 272
  - table
    - creating 37
    - data written to tables 348
    - logical data groups 348
  - table space
    - data written to tables 352
    - logical data groups 352
  - tables
    - managing 89

- event monitors *(continued)*
  - tables *(continued)*
    - pruning 126
    - relationship to logical data groups 91
  - threshold violation
    - logical data groups 342
  - threshold violations
    - data written to tables 342
  - transaction
    - data written to tables 360
  - unformatted event table 120
    - creating 96
  - unformatted event tables
    - methods for accessing data 120
    - routines for extracting data 125
  - unit of work
    - data written to tables 166
    - logical data groups 166
    - overview 163
    - usage example 211
  - upgrading tables 403
  - usage
    - enabling data collection 117
    - methods for accessing event monitor data 118
    - overview 32
  - XML data 269
- event monitorswrite-to-table
  - tabletable 36
- events
  - captured by event monitors 27
  - enabling event monitor data collection
    - overview 117
  - monitor elements
    - event\_time 812
    - start\_time 1242
    - stop\_time 1263
- EVMON\_FORMAT\_UE\_TO\_TABLES procedure
  - PRUNE\_UE\_TABLE option 126
- evmon\_wait\_time monitor element 815
- evmon\_waits\_total monitor element 817
- EVMONSTART
  - change history event monitor
    - logical data groups 382
- examples
  - DB2 pureScale instances
    - viewing status 1447
  - DB2\_INSTANCE\_ALERTS administrative view 1454
  - db2cluster command 1454
  - ENV\_CF\_SYS\_RESOURCES administrative view
    - viewing cluster caching facility processor load 1464
  - MON\_GET\_CF table function
    - viewing cluster caching facility memory usage 1461
  - MON\_GET\_PAGE\_ACCESS\_INFO table function
    - viewing page reclaim statistics 1479
  - MON\_GET\_PKG\_CACHE\_STMT table function
    - viewing statements causing frequent page reclaiming 1479
- monitoring
  - calculating CPU time used by applications or workloads 211
  - capturing activities associated with an SQL statement 270
  - changes performed by the STMM 402
  - identifying candidate statements for performance tuning 246
  - identifying configuration changes 398
  - identifying utility executions 398

- examples *(continued)*
  - monitoring *(continued)*
    - investigating an increase in lock escalations 396
    - listing changes performed by the STMM 402
    - listing committed DDL statements 401
    - LOAD operations 399
    - unit of work event monitor 211
    - using db2advis and package cache information to improve performance 248
    - using the change history event monitor 396
    - utility execution 400
- executable IDs
  - unit of work event monitor 206
- executable lists
  - unit of work event monitor 206

## F

- FCM
  - monitor elements
    - buff\_auto\_tuning 695
    - buff\_free 695
    - buff\_free\_bottom 695
    - buff\_max 696
    - buff\_total 696
    - ch\_auto\_tuning 707
    - ch\_free 707
    - ch\_free\_bottom 707
    - ch\_max 708
    - ch\_total 708
    - fcm\_congested\_sends 823
    - fcm\_congestion\_time 823
    - fcm\_message\_recv\_volume 824
    - fcm\_message\_recv\_wait\_time 825
    - fcm\_num\_congestion\_timeouts 823
    - fcm\_num\_conn\_lost 823
    - fcm\_num\_conn\_timeouts 824
    - hostname 871
    - remote\_member 1176
    - total\_buffers\_rcvd 1319
    - total\_buffers\_sent 1320
  - monitoring 13
  - wait time monitor elements 537
- federated server monitor elements
  - disconnects 804
- fetching
  - fetch\_count monitor element 848
- file event monitors
  - buffering 107
  - creating 101
  - formatting output from command line 127
  - managing 103
- file systems
  - db.log\_fs\_util health indicator 476
  - monitor elements
    - fs\_caching 850
    - fs\_id 851
    - fs\_total\_size 851
    - fs\_used\_size 852
- files
  - files\_closed monitor element 849
- formulas
  - buffer pool hit ratios 1469

## G

- GBPs
  - monitor elements 1465
  - relationship to local buffer pools 1467
- GET SNAPSHOT command
  - sample output 428
- GLMs
  - overview 1475
- global health snapshots 498
- global lock managers
  - overview 1475
- global snapshots on partitioned database systems 430
- global variables
  - monitor elements
    - mon\_interval\_id 968
- group buffer pools
  - monitor elements 1465
    - object\_data\_gbp\_l\_reads 985
    - object\_data\_gbp\_p\_reads 986
    - object\_index\_gbp\_invalid\_pages 989
    - object\_index\_gbp\_l\_reads 989
    - object\_index\_gbp\_p\_reads 990
    - object\_xda\_gbp\_invalid\_pages 994
    - object\_xda\_gbp\_l\_reads 995
    - object\_xda\_gbp\_p\_reads 995
  - relationship to local buffer pools 1467
- gw\_db\_alias element 855

## H

- HADR
  - health indicators
    - db.hadr\_delay 475
    - db.hadr\_op\_status 475
  - monitor elements
    - hadr\_connect\_status 856
    - hadr\_connect\_time 857
    - hadr\_heartbeat 858
    - hadr\_local\_host 858
    - hadr\_local\_service 859
    - hadr\_log\_gap 859
    - hadr\_peer\_window 860
    - hadr\_peer\_window\_end 860
    - hadr\_primary\_log\_file 861
    - hadr\_primary\_log\_lsn 861
    - hadr\_primary\_log\_page 862
    - hadr\_remote\_host 862
    - hadr\_remote\_instance 863
    - hadr\_remote\_service 863
    - hadr\_role 864
    - hadr\_standby\_log\_file 864
    - hadr\_standby\_log\_lsn 865
    - hadr\_standby\_log\_page 865
    - hadr\_state 865
    - hadr\_syncmode 866
    - hadr\_timeout 867
- hash joins
  - monitor elements
    - active\_hash\_joins 644
    - hash\_join\_overflows 868
    - hash\_join\_small\_overflows 868
    - post\_shrthreshold\_hash\_joins 1144
    - post\_threshold\_hash\_joins 1146
    - total\_hash\_joins 1339
- health alerts
  - enabling 488
- health alerts (*continued*)
  - recommendations 499
  - resolving
    - client applications 502
    - SQL queries 499
- health indicators
  - alert actions 509
  - alerts
    - resolving using SQL 499
    - retrieving recommendations 499, 502
  - applications waiting on locks 480
  - catalog cache hit ratio 480
  - collection state-based 455
  - configuring
    - client applications 507
    - overview 503
    - resetting 507
    - retrieving 505
    - updates 506
  - data 491
  - databases
    - heap utilization 482
    - highest severity alert state 472
    - operational state 472
  - db.alert\_state 472
  - db.apps\_waiting\_locks 480
  - db.catcache\_hitratio 480
  - db.db\_auto\_storage\_util 463
  - db.db\_backup\_req 474
  - db.db\_heap\_util 482
  - db.db\_op\_status 472
  - db.deadlock\_rate 477
  - db.fed\_nicknames\_op\_status 483
  - db.fed\_servers\_op\_status 483
  - db.hadr\_delay 475
  - db.hadr\_op\_status 475
  - db.lock\_escal\_rate 479
  - db.locklist\_utilization 478
  - db.log\_fs\_util 476
  - db.log\_util 476
  - db.max\_sort\_shrmem\_util 470
  - db.pkgcache\_hitratio 481
  - db.shrworkspace\_hitratio 481
  - db.sort\_shrmem\_util 468
  - db.spilled\_sorts 469
  - db.tb\_reorg\_req 473
  - db.tb\_runstats\_req 474
  - db2.db2\_alert\_state 471
  - db2.db2\_op\_status 471
  - db2.mon\_heap\_util 482
  - db2.sort\_privmem\_util 468
  - deadlock rate 477
  - DMS table spaces 462
  - format 459
  - instances
    - highest severity alert state 471
    - operational state 471
  - lock escalation rate 479
  - lock list utilization 478
  - logs
    - file system utilization 476
    - space utilization 476
  - monitor heap utilization 482
  - overview 455
  - package cache hit ratio 481
  - process cycle 458
  - shared workspace hit ratio 481

- health indicators *(continued)*
  - sort memory utilization
    - long-term shared 470
    - private 468
    - shared 468
  - sorts that overflowed 469
  - state-based 455
  - summary 459
  - table spaces
    - container operational state 467
    - container utilization 466
    - operational state 467
    - storage utilization 465
  - threshold-based 455
  - ts.ts\_auto\_resize\_status 464
  - ts.ts\_op\_status 467
  - ts.ts\_util 465
  - ts.ts\_util\_auto\_resize 464
  - tsc.tscont\_op\_status 467
  - tsc.utilization 466
- health monitor
  - alerts 503
  - API request types 487
  - CLP commands 486
  - details 455
  - interfaces 484
  - logical data groups 487
  - recommendation retrieval
    - using client application 502
    - using CLP 499
    - using SQL 499
  - sample output 496
  - SQL table functions 485
  - starting 490
  - stopping 490
  - thresholds 503
- health snapshots
  - capturing
    - using client applications 493
    - using CLP 493
    - using SQL table functions 492
  - global 498
- help
  - SQL statements 1552
- histograms
  - monitor elements
    - histogram\_type 869
    - number\_in\_bin 984
    - top 1312
- host databases
  - host\_db\_name monitor element 871
  - name monitor element 871
- hosts
  - DB2 pureScale environments
    - alerts 1439, 1441
    - states 1439, 1441
  - DB2 pureScale instances
    - viewing status 1447

## I

- I/O
  - monitor elements
    - num\_log\_part\_page\_io 978
    - num\_log\_read\_io 979
    - num\_log\_write\_io 979
    - num\_pages\_from\_block\_IOs 1012

## I/O *(continued)*

- monitor elements *(continued)*
  - num\_pages\_from\_vectored\_IOs 1013
  - vectored\_ios 1420
- identifiers
  - monitor elements
    - arm\_correlator 677
    - bin\_id 690
    - db\_work\_action\_set\_id 778
    - db\_work\_class\_id 779
    - host\_prdid 872
    - sc\_work\_action\_set\_id 1199
    - sc\_work\_class\_id 1200
    - service\_class\_id 1208
    - sql\_req\_id 1229
    - work\_action\_set\_id 1425
    - work\_class\_id 1426
  - index\_name monitor element 876
  - index\_schema monitor element 876
- indexes
  - index object pages monitor element 876
  - monitor elements
    - iid 874
    - index\_name 876
    - index\_object\_pages 876
    - index\_only\_scans 877
    - index\_scans 877
    - index\_schema 876
    - index\_tbsp\_id 878
    - int\_node\_splits 883
    - nleaf 970
    - nlevels 971
    - page\_allocations 1010
    - pages\_merged 1013
    - reorg\_index\_id monitor 1178
    - root\_node\_splits 1188
- inline storage
  - LOBs
    - unformatted event tables 96
- insert\_timestamp monitor element 879
- inserting data
  - appl\_section\_inserts monitor element 672
- instances
  - operational state health indicator 471
- int\_rows\_deleted monitor element 885
- invalid pages
  - DB2 pureScale environments 1467
- isolation levels
  - effective\_isolation monitor element 806

## J

- Java tools
  - db2evmonfmt 243

## L

- large objects (LOBs)
  - lob\_object\_pages element 899
- latch waits
  - total\_extended\_latch\_wait\_time monitor element 1336
  - total\_extended\_latch\_waits monitor element 1338
- LBPs
  - monitor elements 1465
  - relationship to group buffer pools 1467

- LIST INSTANCE command
  - overview 1437
- LIST TABLESPACE CONTAINERS command
  - DB2 pureScale environment 1483
- LIST TABLESPACES command
  - DB2 pureScale environment 1483
- LLMs
  - overview 1475
- local buffer pools
  - monitor elements 1465
    - object\_data\_lbp\_pages\_found 987
    - object\_index\_lbp\_pages\_found 990
    - object\_xda\_lbp\_pages\_found 996
  - relationship to group buffer pools 1467
- local lock managers
  - overview 1475
- location monitor element 774
- lock escalation
  - db.lock\_escal\_rate health indicator 479
  - lock\_escalation monitor element 905
- lock list utilization health indicator 478
- lock modes
  - monitor elements
    - lock\_current\_mode 904
    - lock\_mode 913
    - lock\_mode\_requested 914
- lock waits
  - DB2 pureScale environments
    - overview 1475
- lock\_escalation monitor element 905
- locking event monitor
  - data returned
    - logical data groups 132
    - table event monitors 132
- locks
  - DB2 pureScale environment
    - lock waits 1475
  - DB2 pureScale environments
    - between members 1475
    - monitoring 1475
    - overview 1475
  - members in DB2 pureScale environments 1475
  - monitor elements
    - agent\_id\_holding\_lock 651
    - appl\_id\_holding\_lk 668
    - effective\_lock\_timeout 807
    - lock\_attributes 902
    - lock\_count 903
    - lock\_escalation 905
    - lock\_escals 906, 1496
    - lock\_hold\_count 912
    - lock\_list\_in\_use 913
    - lock\_name 915
    - lock\_node 916
    - lock\_object\_name 916
    - lock\_object\_type 917
    - lock\_release\_flags 919
    - lock\_status 920
    - lock\_timeout\_val 921
    - lock\_timeouts 921
    - lock\_wait\_time 925
    - lock\_waits 929
    - locks\_held 933
    - locks\_held\_top 933
    - locks\_in\_list 934
    - locks\_waiting 934
    - participant\_no\_holding\_lk 1016
- locks (*continued*)
  - monitor elements (*continued*)
    - remote\_lock\_time 1176
    - remote\_locks 1176
    - sequence\_no\_holding\_lk 1205
    - stmt\_lock\_timeout 1250
    - uow\_lock\_wait\_time 1404
    - x\_lock\_escals 1430
  - monitoring 12
  - timeouts
    - reports 160
  - waits
    - lock\_wait\_start\_time monitor element 924
    - reports 160
- log buffers
  - num\_log\_buffer\_full monitor element 976
- log disk monitor elements
  - log\_disk\_wait\_time 936
  - log\_disk\_waits\_total 937
- log sequence numbers (LSNs)
  - monitor elements
    - hadr\_primary\_log\_lsn 861
    - hadr\_standby\_log\_lsn 865
- logical data groups
  - buffer pool event monitor 350
  - change history event monitor 367
  - COLLECT ACTIVITY DATA settings effects 599
  - connections event monitor 354
  - data organization 444
  - database event monitor 334
  - event monitors
    - changing 128
    - listing 40, 547
  - health monitor 487
  - lock event monitor 132
  - mapping to event types 110, 596
  - overview 547
  - package cache event monitor 218
  - relationship to event monitor tables 91
  - snapshot monitor 599
  - statement event monitor 343
  - table event monitor 348
  - table space event monitor 352
  - threshold violation event monitor 342
  - unit of work event monitor 166
- logs
  - health indicators
    - db.log\_fs\_util 476
    - db.log\_util 476
  - monitor elements
    - current\_active\_log 766
    - current\_archive\_log 766
    - diaglog\_write\_wait\_time 789
    - diaglog\_writes\_total 790
    - first\_active\_log 849
    - hadr\_log\_gap 859
    - hadr\_primary\_log\_file 861
    - hadr\_primary\_log\_page 862
    - hadr\_standby\_log\_file 864
    - hadr\_standby\_log\_page 865
    - last\_active\_log 894
    - log\_held\_by\_dirty\_pages 938
    - log\_read\_time 939
    - log\_reads 939
    - log\_to\_redo\_for\_recovery 940
    - log\_write\_time 941
    - log\_writes 941

- logs *(continued)*
  - monitor elements *(continued)*
    - sec\_log\_used\_top 1200
    - sec\_logs\_allocated 1201
    - smallest\_log\_avail\_node 1224
    - tot\_log\_used\_top 1312
    - total\_log\_available 1346
    - total\_log\_used 1347
    - uow\_log\_space\_used 1404
- long data
  - long\_object\_pages monitor element 942

## M

- member restart
  - status checking 1452
- members
  - alerts
    - interpreting 1441
    - values 1439
  - monitor elements
    - member 958, 1506
  - restarting
    - status checking 1452
  - states
    - interpreting 1441
    - values 1439
  - status viewing 1448
- memory
  - cluster caching facilities
    - monitor elements 1460
    - monitoring 1459
    - usage 1461
  - health indicators
    - db.sort\_shrmem\_util 468
    - db2.sort\_privmem\_util 468
  - monitor elements
    - comm\_private\_mem 721
    - db\_heap\_top 774
    - lock\_list\_in\_use 913
    - pool\_config\_size 1041
    - pool\_cur\_size 1042
    - pool\_id 1072
    - pool\_secondary\_id 1115
    - pool\_watermark 1128
  - monitoring
    - DB2 pureScale environment 1460
    - overview 13
  - requirements
    - database system monitor 447
- messages
  - monitor elements
    - message 967
    - message\_time 968
- metrics
  - See also* monitor elements
  - activities 521
  - data objects 523
  - ranking monitor elements returned in XML documents 22
  - requests 519
  - returned by event monitors 13
  - system
    - capturing 272
- minimum channels free monitor element 707
- mkfifo command 105
- MON\_FORMAT\_ table functions
  - comparison with XMLTABLE table function 18

- MON\_FORMAT\_ table functions *(continued)*
  - viewing monitor elements as rows in a table 22
- MON\_GET\_CF table function
  - examples 1461
- MON\_GET\_PAGE\_ACCESS\_INFO table function
  - examples 1479
- MON\_GET\_PKG\_CACHE\_STMT table function
  - examples 1479
- mon\_heap\_sz database manager configuration parameter
  - overview 447
- mon\_interval\_id monitor element 968
- monitor element collection levels
  - overview 525
- monitor elements
  - activity\_metrics 280
  - cached\_timestamp 697
  - concurrentdbcoordactivities\_db\_threshold\_value 728
  - concurrentdbcoordactivities\_db\_threshold\_violated 729
  - concurrentdbcoordactivities\_subclass\_threshold\_id 729
  - contained in XML documents
    - formatting 22
  - locks
    - written from EVMON\_FORMAT\_UE\_TO\_TABLES procedure 137
    - written from EVMON\_FORMAT\_UE\_TO\_XML table function 142
  - metrics
    - ranking 22
  - package cache
    - written from EVMON\_FORMAT\_UE\_TO\_TABLES procedure 226
    - written from EVMON\_FORMAT\_UE\_TO\_XML table function 235
  - port\_number 1144
  - priority 1157
  - ssl\_port\_number 1241
  - system\_metrics 280
  - units of work (UOW)
    - written from EVMON\_FORMAT\_UE\_TO\_TABLES procedure 177
    - written from EVMON\_FORMAT\_UE\_TO\_XML table function 189
    - viewing as rows in a table 22
- monitor heap health indicator 482
- monitor switches
  - details 438
  - setting
    - client applications 442
    - CLP 440
- monitoring
  - activities 253
  - activity event monitors 269
  - API request types 425
  - buffer pools 449
    - DB2 pureScale (overview) 1465
    - DB2 pureScale environments (hit rates) 1467
    - DB2 pureScale environments (hit ratios) 1467
  - changes 366
  - CLP commands 422
  - database events 27
  - databases 3
  - DB2 pureScale environments
    - buffer pool hit rates 1467
    - buffer pool hit ratios 1467
  - databases 1457
  - events 1457
  - locking 1475

- monitoring (*continued*)
  - DB2 pureScale environments (*continued*)
    - overview 1435, 1437
    - systems 1457
  - DB2 pureScale instances
    - status 1437
  - db2top command 433
  - events 129
    - change history 396
    - changes performed by the STMM 402
    - committed DDL statements 401
    - configuration changes 398
    - LOAD operations 399
    - units of work 211
    - utility execution 400
    - utility executions 398
  - extent movement status
    - table functions 13
  - FCM (fast communications manager)
    - table functions 13
  - health monitor 455, 490
  - history changes 366
  - interfaces 1
  - locking 129
    - event monitors 129
    - table functions 12
  - locks
    - DB2 pureScale environments 1475
  - monitor data returned in XML documents 13
  - object usage
    - objects that a statement affects 10
    - overview 8
    - statements that affect a table 8
  - package cache eviction events 216
  - page reclaim statistics
    - examples 1479
    - overview 1478
  - snapshot access
    - snapshot table functions in SQL queries 416
    - SYSMON authority 411
  - snapshot capture methods
    - client applications 424
    - CLP 421
    - SNAP\_WRITE\_FILE stored procedure 414
    - snapshot administrative views 412
    - snapshot table functions 412
    - snapshot table functions in SQL queries 416
    - SQL 420
  - statistics 272
  - system catalog views directly
    - last referenced date 453
  - table functions 5
  - unformatted event table 98
  - unit of work events 163
  - utility history 393
- monitoring data 405
- monitoring events
  - change history
    - changes performed by the STMM example 402
    - committed DDL statements example 401
    - configuration changes example 398
    - LOAD operations example 399
    - usage example 396
    - utility execution example 400
    - utility executions example 398
- monitoring tools
  - deprecated 455

- MONREPORT module
  - customizing 408
- MONREPORT reports 405
- most recent response time for connect monitor element 725

## N

- named pipes
  - Linux and UNIX
    - creating 105
- names
  - monitor elements
    - db\_name 774, 1492
    - dcs\_db\_name 781
    - service\_subclass\_name 1209
    - service\_superclass\_name 1210
    - work\_action\_set\_name 1425
    - work\_class\_name 1426
- network time
  - monitor elements
    - max\_network\_time\_1\_ms 957
    - max\_network\_time\_100\_ms 956
    - max\_network\_time\_16\_ms 956
    - max\_network\_time\_4\_ms 957
    - max\_network\_time\_500\_ms 958
    - max\_network\_time\_gt500\_ms 958
    - network\_time\_bottom 969
    - network\_time\_top 970
- nicknames
  - health indicator 483
  - monitor elements
    - create\_nickname 763
    - create\_nickname\_time 764
- nodes
  - monitor elements
    - coord\_node 750
    - node\_number 971
    - num\_nodes\_in\_db2\_instance 981
    - ss\_node\_number 1239
- notices 1559
- num\_indoubt\_trans element 976
- num\_transmissions element 983
- num\_transmissions\_group element 984
- numbers
  - monitor elements
    - progress\_list\_cur\_seq\_num 1161
    - ss\_number 1239

## O

- object\_data\_gbp\_invalid\_pages monitor element 985
- object\_data\_gbp\_l\_reads monitor element 985
- object\_data\_gbp\_p\_reads monitor element 986
- object\_data\_l\_reads monitor element 987
- object\_data\_lbp\_pages\_found monitor element 987
- object\_data\_p\_reads monitor element 988
- object\_index\_gbp\_invalid\_pages monitor element 989
- object\_index\_gbp\_l\_reads monitor element 989
- object\_index\_gbp\_p\_reads monitor element 990
- object\_index\_l\_reads monitor element 991
- object\_index\_lbp\_pages\_found monitor element 990
- object\_index\_p\_reads monitor element 992
- object\_name monitor element 992
- object\_schema monitor element 993
- object\_xda\_gbp\_invalid\_pages monitor element 994
- object\_xda\_gbp\_l\_reads monitor element 995

- object\_xda\_gbp\_p\_reads monitor element 995
- object\_xda\_l\_reads monitor element 996
- object\_xda\_lbp\_pages\_found monitor element 996
- object\_xda\_p\_reads monitor element 997
- objects
  - monitor elements
    - object\_data\_gbp\_invalid\_pages 985
    - object\_name 992
  - monitoring
    - object usage 8
    - objects that a statement affects 10
    - statements that affect a table 8
  - performance (Windows) 514
  - usage 8
- OLAP
  - monitor elements
    - active\_olap\_funcs 644
    - olap\_func\_overflows 998
    - post\_threshold\_olap\_funcs 1147
    - total\_olap\_funcs 1347
- operation monitor element 1251
- operations
  - monitor elements
    - direct\_read\_reqs 791
    - direct\_read\_time 793
    - direct\_reads 795
    - direct\_write\_reqs 797
    - direct\_write\_time 799
    - direct\_writes 801
    - stmt\_operation 1251
- optimization
  - monitor elements
    - stmt\_value\_isreopt 1262
- outbound bytes received
  - monitor elements
    - max\_data\_received\_1024 947
    - max\_data\_received\_128 947
    - max\_data\_received\_16384 947
    - max\_data\_received\_2048 948
    - max\_data\_received\_256 948
    - max\_data\_received\_31999 949
    - max\_data\_received\_4096 949
    - max\_data\_received\_512 949
    - max\_data\_received\_64000 950
    - max\_data\_received\_8192 950
    - max\_data\_received\_gt64000 951
    - outbound\_bytes\_received 1002
    - outbound\_bytes\_received\_bottom 1003
    - outbound\_bytes\_received\_top 1003
- outbound bytes sent
  - monitor elements
    - max\_data\_sent\_1024 951
    - max\_data\_sent\_128 952
    - max\_data\_sent\_16384 952
    - max\_data\_sent\_2048 952
    - max\_data\_sent\_256 953
    - max\_data\_sent\_31999 953
    - max\_data\_sent\_4096 954
    - max\_data\_sent\_512 954
    - max\_data\_sent\_64000 955
    - max\_data\_sent\_8192 955
    - max\_data\_sent\_gt64000 955
    - outbound\_bytes\_sent 1004
    - outbound\_bytes\_sent\_bottom 1004
    - outbound\_bytes\_sent\_top 1004

- outbound communication
  - monitor elements
    - outbound\_appl\_id 1002
    - outbound\_comm\_address 1004
    - outbound\_comm\_protocol 1005
    - outbound\_sequence\_no 1005
- overflow records
  - event monitors 89
  - monitor elements
    - first\_overflow\_time 850
    - last\_overflow\_time 896
    - overflow\_accesses 1005
    - overflow\_creates 1006

## P

- package cache
  - db.pkgcache\_hitratio health indicator 481
  - monitor elements
    - pkg\_cache\_inserts 1021
    - pkg\_cache\_lookups 1022
    - pkg\_cache\_num\_overflow 1024
    - pkg\_cache\_size\_top 1024
- package cache event monitor
  - data returned
    - logical data groups 218
    - table event monitors 219
  - monitor data returned in XML documents 13
  - overview 216
  - usage example
    - improving database performance 248
    - tuning statements 246
- package cache event monitor reports 243
- package listing
  - unit of work event monitor 201
- packages
  - monitor elements
    - package\_name 1007
    - package\_schema 1008
    - package\_version\_id 1009
    - stmt\_pkgcache\_id 1252
- page reclaiming
  - monitor data
    - viewing 1479
  - overview 1478
- page size
  - unformatted event tables 96
- page validity
  - DB2 pureScale environments 1467
- pages
  - bp\_pages\_left\_to\_remove monitor element 694
  - data\_object\_pages monitor element 767
  - removing 694
  - sizesunformatted event tables 96
- parallelism
  - monitor elements
    - degree\_parallelism 786
- partial\_record monitor element 1015
- partition\_number monitor element 1018
- partitioned databases 114
  - global snapshots 430
  - monitor elements
    - coord\_partition\_num 751
- pass-through monitor elements
  - passthru\_time 1018
  - passthru 1018

- performance
  - db2advis
    - creating input file with the package cache event monitor 248
  - identifying costly statements from the package cache 246
  - identifying statements that affect tables 8
  - information
    - displaying 514
    - enabling remote access 513
  - remote databases 515
  - resetting values 515
  - SQL query
    - using object statistics 10
  - time-spent monitor elements 529
  - Windows
    - monitoring tools 512
    - Performance Monitor objects 514
- pipe event monitors
  - creating 105
  - formatting output from command line 127
  - named pipe management 106
- pipeds\_sorts\_accepted monitor element 1020
- pipeds\_sorts\_requested monitor element 1020
- pool\_async\_data\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1025
- pool\_async\_index\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1030
- pool\_async\_xda\_gbp\_indep\_pages\_found\_in\_lbp monitor element 1036
- post\_shrthreshold\_sorts monitor element 1145
- prefetching
  - unread\_prefetch\_page monitor elements 1400
- priv\_workspace\_num\_overflows monitor element 1157
- priv\_workspace\_section\_inserts monitor element 1158
- priv\_workspace\_section\_lookups monitor element 1159
- priv\_workspace\_size\_top monitor element 1159
- problem determination
  - information available 1556
  - tutorials 1556
- processes
  - monitor elements
    - agent\_pid 652
- processor utilization
  - monitor elements
    - cpu\_idle 754
    - cpu\_iowait 755
    - cpu\_system 757
    - cpu\_usage\_total 759
    - cpu\_user 759
- processors
  - cluster caching facilities
    - load monitoring 1459
    - viewing load 1464
- progress\_description monitor element 1161
- progress\_seq\_num monitor element 1162
- progress\_start\_time monitor element 1162
- progress\_work\_metric monitor element 1162
- pruning event monitor data 126

## Q

- queries
  - monitor elements
    - query\_card\_estimate 1164
    - query\_cost\_estimate 1164
    - query\_data\_tag\_list 1165
    - queue\_assignments\_total 1166

- queries (*continued*)
  - monitor elements (*continued*)
    - queue\_size\_top 1166
    - queue\_time\_total 1167
    - select\_time 1204
- quiescer
  - monitor elements
    - quiescer\_auth\_id 1168
    - quiescer\_obj\_id 1168
    - quiescer\_state 1168
    - quiescer\_ts\_id 1169

## R

- range adjustment monitor element 1169
- range container monitor element 1169
- range number monitor element 1170
- range offset monitor element 1170
- range\_num\_containers monitor element 1170
- ranges
  - monitor elements
    - bottom 692
    - range\_adjustment 1169
    - range\_container\_id 1169
    - range\_end\_stripe 1169
    - range\_max\_extent 1170
    - range\_max\_page\_number 1170
    - range\_num\_containers 1170
    - range\_number 1170
    - range\_offset 1170
    - range\_start\_stripe 1171
    - range\_stripe\_set\_number 1171
- ranking monitor elements 22
- real-time statistics
  - monitor elements
    - stats\_fabricate\_time 1244
    - stats\_fabrications 1245
- rebalancing
  - monitor elements
    - current\_extent 767
    - rebalancer\_extents\_processed 1285
    - rebalancer\_extents\_remaining 1285
    - rebalancer\_last\_extent\_moved 1286
    - rebalancer\_mode 1286
    - rebalancer\_priority 1287
    - rebalancer\_restart\_time 1288
    - rebalancer\_start\_time 1289
    - rebalancer\_status 1289
    - rebalancer\_target\_storage
      - \_group\_id 1290
    - rebalancer\_target\_storage
      - \_group\_name 1290
    - tablespace\_rebalancer\_extents\_processed 1285
    - tablespace\_rebalancer\_extents\_remaining 1285
    - tablespace\_rebalancer\_last\_extent\_moved 1286
    - tablespace\_rebalancer\_mode 1286
    - tablespace\_rebalancer\_priority 1287
    - tablespace\_rebalancer\_restart\_time 1288
    - tablespace\_rebalancer\_source\_storage\_group\_id 1288
    - tablespace\_rebalancer\_source\_storage
      - \_group\_name 1288
    - tablespace\_rebalancer\_start\_time 1289
    - tablespace\_rebalancer\_status 1289
    - tablespace\_rebalancer\_target\_storage
      - \_group\_id 1290
    - tablespace\_rebalancer\_target\_storage
      - \_group\_name 1290

- rebalancing *(continued)*
  - monitor elements *(continued)*
    - tbsp\_rebalancer\_extents\_processed 1285
    - tbsp\_rebalancer\_extents\_remaining 1285
    - tbsp\_rebalancer\_last\_extent\_moved 1286
    - tbsp\_rebalancer\_mode 1286
    - tbsp\_rebalancer\_priority 1287
    - tbsp\_rebalancer\_restart\_time 1288
    - tbsp\_rebalancer\_start\_time 1289
    - tbsp\_rebalancer\_status 1289
    - tbsp\_rebalancer\_target\_storage
      - \_group\_name 1290
    - tbsp\_target\_storage
      - \_group\_id 1290
- rebinding
  - monitor elements
    - int\_auto\_rebinds 880
- reclaiming
  - pages in use by other members 1478
- records
  - monitor elements
    - partial\_record 1015
- recovery
  - monitor elements
    - log\_to\_redo\_for\_recovery 940
- REGVAR
  - change history event monitor
    - logical data groups 377
- remote databases
  - performance information 515
- reoptimization monitor elements
  - stmt\_value\_isreopt 1262
- reorg\_index\_id monitor element 1178
- reorganization
  - health indicators
    - db.tb\_reorg\_req 473
  - monitor elements
    - page\_reorgs 1010
    - reorg\_current\_counter 1177
    - reorg\_max\_counter 1178
    - reorg\_max\_phase 1179
    - reorg\_phase 1179
    - reorg\_phase\_start 1180
    - reorg\_rows\_compressed 1180
    - reorg\_rows\_rejected\_for\_compression 1180
    - reorg\_start 1181
    - reorg\_status 1181
    - reorg\_type 1181
- reorganize phase monitor element 1179
- reports
  - change history 395
  - deadlock 160
  - lock timeouts 160
  - lock waits 160
  - package cache 243
  - units of work 209
- request identifier for sql statement monitor element 1229
- request metrics
  - See request monitor elements 519
- request monitor elements
  - overview 519
  - rqsts\_completed\_total 1198
- requests
  - monitoring 519
- response times
  - monitor elements
    - delete\_time 787
- response times *(continued)*
  - monitor elements *(continued)*
    - host\_response\_time 873
    - insert\_time 879
- restart status
  - member 1452
- rollbacks
  - monitor elements
    - int\_deadlock\_rollbacks 882
    - int\_rollbacks 883
    - rf\_status 1185
    - rollback\_sql\_stmts 1186
    - rolled\_back\_agent\_id 1186
    - rolled\_back\_appl\_id 1187
    - rolled\_back\_participant\_no 1187
    - rolled\_back\_sequence\_no 1187
- rollforward recovery
  - monitor elements
    - rf\_log\_num 1184
    - rf\_status 1185
    - rf\_timestamp 1185
    - rf\_type 1185
    - tablespace\_min\_recovery\_time 1280, 1544
    - tbsp\_min\_recovery\_time 1280, 1544
    - ts\_name 1398
- routine\_id monitor element 1188
- routines
  - monitor elements
    - routine\_id 1188
- row-based formatting functions 22
- rows
  - monitor elements
    - int\_rows\_inserted 885
    - int\_rows\_updated 886
    - rows\_deleted 1189
    - rows\_fetched 1189
    - rows\_inserted 1190
    - rows\_modified 1191
    - rows\_read 1192
    - rows\_returned 1194
    - rows\_returned\_top 1195
    - rows\_selected 1196
    - rows\_updated 1197
    - rows\_written 1197
    - sp\_rows\_selected 1228
  - rows compressed monitor element 1180
  - rows rejected for compression
    - monitor element 1180
  - rows returned by stored procedures
    - monitor element 1228
  - rows selected monitor element 1196
- RUNSTATS utility
  - monitor elements
    - async\_runstats 678
    - sync\_runstats 1266
    - sync\_runstats\_time 1267

## S

- schemas
  - monitor elements
    - object\_schema 993
    - table\_schema monitor element 1272, 1542
- sections
  - monitor elements
    - appl\_section\_inserts 672
    - appl\_section\_lookups 673

- sections (*continued*)
  - monitor elements (*continued*)
    - priv\_workspace\_section\_inserts 1158
    - priv\_workspace\_section\_lookups 1159
    - section\_env 1202
    - section\_number 1202
  - select SQL statements executed monitor element 1204
- self-describing data streams
  - database system monitor 446
  - event monitors 108
  - snapshot monitor 431
  - system monitor switches 443
- sequences
  - monitor elements
    - progress\_seq\_num 1162
    - sequence\_no 1205
    - sequence\_no\_holding\_lk 1205
- servers
  - monitor elements
    - product\_name 1160
    - server\_instance\_name 1206
    - server\_platform 1207
    - server\_prdid 1207
    - server\_version 1208
- service-level information
  - service\_level monitor element 1209
- session authorization IDs
  - monitor element 1212
- shared workspaces
  - health indicators
    - db.shrworkspace\_hitratio 481
  - monitor elements
    - shr\_workspace\_num\_overflows 1213
    - shr\_workspace\_section\_inserts 1213
    - shr\_workspace\_section\_lookups 1214
    - shr\_workspace\_size\_top 1214
- snapshot monitoring
  - API request types 425
  - capturing snapshots
    - to file 414
    - using SQL with file access 416
  - CLP commands 422
  - locking
    - DB2 pureScale environment 1483
  - making snapshot data available for all users 414
  - methods
    - client applications 424
    - CLP 421
    - SNAP\_WRITE\_FILE stored procedure 414
    - SQL 420
    - SQL with direct access 412
  - output
    - samples 428
    - self-describing data streams 431
  - overview 410
  - partitioned database systems 430
  - request types 422
  - SQL table functions 417
  - subsections 429
- snapshot time monitor element 1311
- snapshots
  - monitor elements
    - time\_stamp 1311
- sort share heap currently allocated monitor element 1227
- sort share heap high watermark monitor element 1227
- sorting
  - health indicators
    - db2.sort\_privmem\_util 468
  - monitor elements
    - active\_sorts 644
    - db.spilled\_sorts 469
    - pipeds\_sorts\_accepted 1020
    - pipeds\_sorts\_requested 1020
    - post\_shrthreshold\_sorts 1145
    - post\_threshold\_sorts 1151
    - sort\_heap\_allocated 1224
    - sort\_heap\_top 1225
    - sort\_overflows 1225
    - sort\_shrheap\_allocated 1227
    - sort\_shrheap\_top 1227
    - total\_sorts 1376, 1385
- SQL
  - operations
    - elapsed\_exec\_time element monitor element 808
  - table functions
    - capturing health snapshots 492
    - health monitor 485
  - SQL requests since last commit monitor element 1229
  - SQL statements
    - help
      - displaying 1552
    - monitor elements
      - ddl\_sql\_stmts 782
      - dynamic\_sql\_stmts 805
      - failed\_sql\_stmts 822
      - insert\_sql\_stmts 878
      - num\_compilation 973
      - num\_executions 974
      - prep\_time\_best 1156
      - prep\_time\_worst 1156
      - select\_sql\_stmts 1204
      - sql\_chains 1229
      - sql\_reqs\_since\_commit 1229
      - sql\_stmts 1230
      - static\_sql\_stmts 1242
      - stmt\_first\_use\_time 1247
      - stmt\_history\_id 1247
      - stmt\_history\_list\_size 1248
      - stmt\_invocation\_id 887, 1248
      - stmt\_isolation 1249
      - stmt\_last\_use\_time 1249
      - stmt\_nest\_level 968, 1250
      - stmt\_node\_number 1251
      - stmt\_pkgcache\_id 1252
      - stmt\_query\_id 1253
      - stmt\_sorts 1254
      - stmt\_source\_id 1254
      - stmt\_text 1256
      - stmt\_type 1257
      - stmt\_value\_data 1260
      - stmt\_value\_index 1260
      - stmt\_value\_isnull 1261
      - stmt\_value\_type 1262
      - total\_exec\_time 1336
      - uid\_sql\_stmts 1399
  - sql\_chains element 1229
  - sql\_stmts element 1230
  - SQLCA
    - monitor elements
      - sqlca 1230

- SQLTEMPSPACE activity threshold
  - monitor elements
    - sqltempespace\_threshold\_id 1234
- ss\_status monitor element 1239
- start stripe monitor element 1171
- statement best preparation time monitor element 1156
- statement concentrator
  - monitor elements
    - eff\_stmt\_txt 806
- statement event monitor
  - data returned
    - logical data groups 343
    - table event monitors 344
- statement first use time monitor element 1247
- statement history identifier monitor element 1247
- statement history list size monitor element 1248
- statement invocation identifier monitor element 887, 1248
- statement isolation monitor element 1249
- statement last use time monitor element 1249
- statement nesting level monitor element 968, 1250
- statement node monitor element 1251
- statement operation monitor element 1251
- statement query identifier monitor element 1253
- statement sorts monitor element 1254
- statement source identifier monitor element 1254
- statement thresholds
  - example 270
- statement type monitor element 1257
- statement worst preparation time monitor element 1156
- statements
  - associated activities 270
  - eviction from package cache 216
  - related activities 270
- states
  - DB2 pureScale environments
    - interpreting 1441
    - values 1439
  - health indicators
    - db.alert\_state 472
    - db.db\_op\_status 472
    - db2.db2\_op\_status 471
    - ts.ts\_op\_status 467
- static SQL statements attempted monitor element 1242
- statistics
  - collection
    - health indicators 474
  - page reclaiming 1478
  - workload management
    - accumulation and resetting 272
- statistics event monitor
  - data returned
    - table event monitors 274
  - monitor data returned in XML documents 13
- status
  - DB2 pureScale instances
    - cluster caching facilities 1448
    - hosts 1447
    - members 1448, 1452
    - overview 1447
    - retrieval interfaces 1437
  - monitor elements
    - appl\_status 673
    - db\_status 776
    - db2\_status 772
    - dcs\_appl\_status 781
    - ss\_status 1239
- stmt\_operation element 1251

- storage paths
  - monitor elements
    - num\_db\_storage\_paths 974
- stored procedure time monitor element 1264
- stored procedures
  - monitor elements
    - stored\_proc\_time 1264
    - stored\_procs 1265
- stored procedures monitor element 1265
- stripe set number monitor element 1171
- stripe sets
  - monitor elements
    - container\_stripe\_set 740
- subsection execution elapsed time monitor element 1238
- subsection node number monitor element 1239
- subsection number monitor element 1239
- subsection status monitor element 1239
- subsections
  - snapshots 429
- SYSCAT.EVENTMONITORS
  - example 113
- SYSCAT.EVENTS
  - example 113
- SYSMON (system monitor) authority
  - details 411
- system metrics
  - accumulation 272
  - capturing with the statistics event monitor 272
  - statistics event monitor
    - data captured 274
- System Monitor Guide and Reference
  - overview xxv
- system monitor switches
  - details 438
  - self-describing data streams 443
  - setting
    - client applications 442
    - CLP 440
  - types 438

## T

- table event monitors
  - data returned
    - logical data groups 348
    - table event monitors 348
  - table management 89
- table functions
  - DB2\_GET\_CLUSTER\_HOST\_STATE
    - overview 1437
  - DB2\_GET\_INSTANCE\_INFO
    - overview 1437
  - monitor 5
  - monitoring
    - activities 6
    - data objects 7
    - extent movement 13
    - FCM (Fast Communications Manager) 13
    - locking 12
    - memory 13
    - miscellaneous 13
    - object usage 8
    - system information 5
- table queues
  - monitor elements
    - tq\_cur\_send\_spills 1391
    - tq\_id\_waiting\_on 1391

table queues (continued)

monitor elements (continued)

tq\_max\_send\_spills 1391  
tq\_node\_waited\_for 1392  
tq\_rows\_read 1392  
tq\_rows\_written 1393  
tq\_tot\_send\_spills 1397  
tq\_wait\_for\_any 1398

table reorganization

monitor elements

reorg\_end 1178  
reorg\_xml\_regions\_compressed 1182  
reorg\_xml\_regions\_rejected\_for\_compression 1182  
table reorganize attribute flag 1181  
table reorganize completion flag 1177  
table reorganize end time 1178  
table reorganize phase start time 1180  
table reorganize start time 1181  
table reorganize status 1181

table reorganize attribute flag monitor element 1181

table reorganize completion flag monitor element 1177

table reorganize end time monitor element 1178

table reorganize phase start time monitor element 1180

table reorganize start time monitor element 1181

table reorganize status monitor element 1181

table space event monitor

data returned

logical data groups 352  
table event monitors 352

table spaces

health indicators

ts.ts\_auto\_resize\_status 464  
ts.ts\_op\_status 467  
ts.ts\_util 465  
ts.ts\_util\_auto\_resize 464  
tsc.tscont\_op\_status 467  
tsc.utilization 466

monitor elements

bp\_tbsp\_use\_count 694  
index\_tbsp\_id 878  
long\_tbsp\_id 943  
quiescer\_ts\_id 1169  
rebalancer\_extents\_processed 1285  
rebalancer\_extents\_remaining 1285  
rebalancer\_last\_extent\_moved 1286  
rebalancer\_mode 1286  
rebalancer\_priority 1287  
rebalancer\_restart\_time 1288  
rebalancer\_start\_time 1289  
rebalancer\_status 1289  
rebalancer\_target\_storage  
\_group\_id 1290  
rebalancer\_target\_storage  
\_group\_name 1290  
reorg\_long\_tbsp\_id 1178  
reorg\_tbsp\_id 1181  
tablespace\_auto\_resize\_enabled 1274  
tablespace\_content\_type 1275  
tablespace\_cur\_pool\_id 1275  
tablespace\_current\_size 1276  
tablespace\_extent\_size 1276  
tablespace\_free\_pages 1276  
tablespace\_id 1277  
tablespace\_increase\_size 1278  
tablespace\_increase\_size\_percent 1278  
tablespace\_initial\_size 1278  
tablespace\_last\_resize\_failed 1279

table spaces (continued)

monitor elements (continued)

tablespace\_last\_resize\_time 1279  
tablespace\_max\_size 1279  
tablespace\_min\_recovery\_time 1280, 1544  
tablespace\_name 1280  
tablespace\_next\_pool\_id 1281  
tablespace\_num\_containers 1282  
tablespace\_num\_quiescers 1282  
tablespace\_num\_ranges 1282  
tablespace\_page\_size 1282  
tablespace\_page\_top 1283  
tablespace\_pending\_free\_pages 1284  
tablespace\_prefetch\_size 1284  
tablespace\_rebalancer\_extents\_processed 1285  
tablespace\_rebalancer\_extents\_remaining 1285  
tablespace\_rebalancer\_last\_extent\_moved 1286  
tablespace\_rebalancer\_mode 1286  
tablespace\_rebalancer\_priority 1287  
tablespace\_rebalancer\_restart\_time 1288  
tablespace\_rebalancer\_source\_storage  
\_group\_id 1288  
tablespace\_rebalancer\_source\_storage  
\_group\_name 1288  
tablespace\_rebalancer\_start\_time 1289  
tablespace\_rebalancer\_status 1289  
tablespace\_rebalancer\_target\_storage  
\_group\_id 1290  
tablespace\_rebalancer\_target\_storage  
\_group\_name 1290  
tablespace\_state 1291  
tablespace\_state\_change\_object\_id 1293  
tablespace\_state\_change\_ts\_id 1293  
tablespace\_total\_pages 1293  
tablespace\_type 1294  
tablespace\_usable\_pages 1294  
tablespace\_used\_pages 1295  
tablespace\_using\_auto\_storage 1295  
tbsp\_auto\_resize\_enabled 1274  
tbsp\_content\_type 1275  
tbsp\_cur\_pool\_id 1275  
tbsp\_current\_size 1276  
tbsp\_datatag 1297  
tbsp\_extent\_size 1276  
tbsp\_free\_pages 1276  
tbsp\_id 1277  
tbsp\_increase\_size 1278  
tbsp\_increase\_size\_percent 1278  
tbsp\_initial\_size 1278  
tbsp\_last\_resize\_failed 1279  
tbsp\_last\_resize\_time 1279  
tbsp\_max\_page\_top 1297  
tbsp\_max\_size 1279  
tbsp\_min\_recovery\_time 1280, 1544  
tbsp\_next\_pool\_id 1281  
tbsp\_num\_containers 1282  
tbsp\_num\_quiescers 1282  
tbsp\_num\_ranges 1282  
tbsp\_page\_size 1282  
tbsp\_page\_top 1283  
tbsp\_pending\_free\_pages 1284  
tbsp\_prefetch\_size 1284  
tbsp\_rebalancer\_extents\_processed 1285  
tbsp\_rebalancer\_extents\_remaining 1285  
tbsp\_rebalancer\_last\_extent\_moved 1286  
tbsp\_rebalancer\_mode 1286  
tbsp\_rebalancer\_priority 1287

table spaces (*continued*)

- monitor elements (*continued*)
  - tbsp\_rebalancer\_restart\_time 1288
  - tbsp\_rebalancer\_start\_time 1289
  - tbsp\_rebalancer\_status 1289
  - tbsp\_rebalancer\_target\_storage\_group\_id 1290
  - tbsp\_rebalancer\_target\_storage\_group\_name 1290
  - tbsp\_state 1291
  - tbsp\_state\_change\_object\_id 1293
  - tbsp\_state\_change\_ts\_id 1293
  - tbsp\_total\_pages 1293
  - tbsp\_trackmod\_state 1298
  - tbsp\_type 1294
  - tbsp\_usable\_pages 1294
  - tbsp\_used\_pages 1295
  - tbsp\_using\_auto\_storage 1295
  - ts\_name 1398

tables

- monitor elements
  - tab\_file\_id 1269
  - tab\_type 1269
  - table\_file\_id 1269
  - table\_name 1270, 1541
  - table\_scans 1272
  - table\_schema 1272, 1542
  - table\_type 1274

target tables

- event monitors 89

TCP/IP

- monitor elements
  - tcip\_sends\_total 1303

terms and conditions

- publications 1556

territory codes

- monitor elements
  - territory\_code 1305

threads

- monitor elements
  - agent\_pid 652

threshold violations event monitor

- data returned
  - logical data groups 342
  - table event monitors 342

thresholds

- health indicators 455
- monitor elements
  - num\_threshold\_violations 983
  - sqltempespace\_threshold\_id 1234
  - thresh\_violations 1305
  - threshold\_action 1307
  - threshold\_domain 1307
  - threshold\_maxvalue 1308
  - threshold\_name 1308
  - threshold\_predicate 1308
  - threshold\_queuesize 1309
  - thresholdid 1310
- statement
  - example 270

time

- monitor elements
  - prefetch\_wait\_time 1152
  - prep\_time 1155
  - progress\_start\_time 1162
  - ss\_exec\_time 1238
  - stmt\_elapsed\_time 1246

time (*continued*)

- monitor elements (*continued*)
  - time\_completed 1310
  - time\_created 1311
  - time\_of\_violation 1311
  - time\_started 1311
  - total\_sort\_time 1375
- time spent
  - monitor elements
    - examples 539
    - hierarchy 531
    - overview 529
    - viewing as rows in a table 22
  - viewing
    - across the system 539
    - during SQL statement execution 543
  - waiting on latches
    - total\_extended\_latch\_wait\_time monitor element 1336
    - total\_extended\_latch\_waits monitor element 1338
- time stamps
  - monitor elements
    - activate\_timestamp 644
    - db\_conn\_time 773
    - db2start\_time 773
    - last\_backup 895
    - last\_reset 897
    - lock\_wait\_start\_time 924
    - message\_time 968
    - prev\_uow\_stop\_time 1156
    - statistics\_timestamp 1242
    - status\_change\_time 1245
    - stmt\_start 1255
    - stmt\_stop 1255
    - uow\_start\_time 1405
    - uow\_stop\_time 1406
  - time waited for prefetch monitor element 1152
  - time zone displacement monitor element 1312
  - time zones
    - time\_zone\_disp element monitor element 1312
- tokens
  - monitor elements
    - consistency\_token 738
    - corr\_token 752
- total completed units of work
  - monitor elements
    - uow\_completed\_total 1401
- total dispatcher queue time
  - monitor elements
    - total\_disp\_run\_queue\_time 1334
- total FCM buffers received monitor element 1319
- total hash loops monitor element 1340
- total log available monitor element 1346
- total log space used monitor element 1347
- total number of pages in object monitor element 1178
- total progress work units monitor element 1162
- total sort time monitor element 1375
- total sorts monitor element 1376, 1385
- transaction event monitor
  - data returned
    - table event monitors 360
  - data written to tablesSee also: unit of work event monitors 360
- transaction processing monitors
  - monitor elements
    - client\_acctng 709
    - client\_applname 710
    - client\_userid 717

- transaction processing monitors *(continued)*
  - monitor elements *(continued)*
    - client\_wrkstnname 718
    - tpmon\_acc\_str 1388
    - tpmon\_client\_app 1389
    - tpmon\_client\_userid 1390
    - tpmon\_client\_wkstn 1390
- transactions
  - monitor elements
    - num\_indoubt\_trans 976
    - xid 1432
- troubleshooting
  - DB2 pureScale instances
    - status monitoring 1437
  - online information 1556
  - SQL 405
  - tutorials 1556
- tutorials
  - list 1555
  - problem determination 1556
  - pureXML 1555
  - troubleshooting 1556
- TXNCOMPLETION
  - change history event monitor
    - logical data groups 381
- type at monitored (server) node monitor element 1206

## U

- UE table unformatted event table 96
- unformatted event tables
  - column definitions 98
  - comparison of data returned in regular tables 100
  - db2evmonfmt Java-based tool for parsing data 120
  - methods for accessing data 120
  - overview 34
  - page size for inline LOBs 96
  - pruning 126
  - routines for extracting data 125
- unit of work event monitor
  - collecting data 209
  - data returned
    - logical data groups 166
    - table event monitors 166
  - monitor data returned in XML documents 13
  - usage example 211
- units of work
  - monitor elements
    - completion\_status 722
    - parent\_uow\_id 1015
    - prev\_uow\_stop\_time 1156
    - progress\_total\_units 1162
    - uow\_comp\_status 1400
    - uow\_completed\_total 1401
    - uow\_elapsed\_time 1402
    - uow\_id 1402
    - uow\_lifetime\_avg 1403
    - uow\_lock\_wait\_time 1404
    - uow\_log\_space\_used 1404
    - uow\_start\_time 1405
    - uow\_status 1406
    - uow\_stop\_time 1406
    - uow\_throughput 1407
  - units of work lifetime average
    - monitor elements
      - uow\_lifetime\_avg 1403
  - units of work throughput
    - monitor elements
      - uow\_throughput 1407
  - update response time monitor element 1408
  - update\_time element 1408
  - update/insert/delete SQL statements executed monitor element 1399
  - updates
    - DB2 Information Center 1552, 1554
    - monitor elements
      - update\_sql\_stmts 1408
    - updates monitor element 1408
  - upgrade
    - event monitor tables
      - detailsimplications of not upgrading 403
  - usage lists
    - monitor elements
      - usage\_list\_last\_state\_change 1409
      - usage\_list\_last\_updated 1409
      - usage\_list\_mem\_size 1409
      - usage\_list\_name 1410
      - usage\_list\_num\_ref\_with\_metrics 1410
      - usage\_list\_num\_references 1410
      - usage\_list\_schema 1411
      - usage\_list\_size 1411
      - usage\_list\_state 1411
      - usage\_list\_used\_entries 1411
      - usage\_list\_wrapped 1412
    - usage\_list\_last\_state\_change monitor element 1409
    - usage\_list\_last\_updated monitor element 1409
    - usage\_list\_mem\_size monitor element 1409
    - usage\_list\_name monitor element 1410
    - usage\_list\_num\_ref\_with\_metrics monitor element 1410
    - usage\_list\_num\_references monitor element 1410
    - usage\_list\_schema monitor element 1411
    - usage\_list\_size monitor element 1411
    - usage\_list\_state monitor element 1411
    - usage\_list\_used\_entries monitor element 1411
    - usage\_list\_wrapped monitor element 1412
    - user authorization level monitor element 688
  - utilities
    - monitor elements
      - utility\_dbname 1412
      - utility\_description 1413
      - utility\_id 1413
      - utility\_invoker\_type 1414
      - utility\_priority 1417
      - utility\_start\_time 1417
      - utility\_state 1417
      - utility\_type 1418
  - utility
    - history
      - monitoring 393
  - UTILLOCATION
    - change history event monitor
      - logical data groups 387
  - UTILPHASE
    - change history event monitor
      - logical data groups 391
  - UTILSTART
    - change history event monitor
      - logical data groups 383
  - UTILSTOP
    - change history event monitor
      - logical data groups 389

## V

- value data monitor element 1260
- value has null value monitor element 1261
- value index monitor element 1260
- value type monitor element 1262
- version monitor element 1420
- virtual memory
  - cluster caching facilities 1459
- virtual storage
  - cluster caching facilities 1459

## W

- wait times
  - monitor elements
    - FCM (Fast Communications Manager) 537
    - overview 531
    - total\_wait\_time 1387
  - viewing
    - activity-level examples 543
    - system-level examples 539
- watermark monitor elements
  - act\_cpu\_time\_top 639
  - act\_rows\_read\_top 641
  - concurrent\_act\_top 725
  - concurrent\_connection\_top 726
  - concurrent\_wlo\_act\_top 726
  - concurrent\_wlo\_top 727
  - coord\_act\_lifetime\_top 746
  - cost\_estimate\_top 752
  - lock\_wait\_time\_top 929
  - rows\_returned\_top 1195
  - temp\_tablespace\_top 1304
  - uow\_total\_time\_top 1407
- Windows
  - Performance Monitor
    - overview 512
    - registering DB2 513
- Windows Management Instrumentation (WMI)
  - DB2 database system integration 511
  - details 510
- workload management
  - monitor element collection levels
    - settings 525
- workload management dispatcher
  - monitor elements
    - cpu\_limit 756
    - cpu\_share\_type 757
    - cpu\_shares 757
    - cpu\_utilization 760
    - cpu\_velocity 760
    - estimated\_cpu\_entitlement 809
    - total\_disp\_run\_queue\_time 1334
- workloads
  - monitor elements
    - wlo\_completed\_total 1425
    - workload\_id 1427
    - workload\_name 1428
    - workload\_occurrence\_id 1429
    - workload\_occurrence\_state 1429
- write-to-table event monitors
  - buffering 107

## X

- XDA Object Pages monitor element 1431
- xda\_object\_pages monitor element 1431
- XML
  - monitor elements
    - formatting 22
    - overview 13
- XML documents
  - monitor elements 13
- XMLTABLE table function
  - comparison with MON\_FORMAT\_ table functions 18
- xquery\_stmts monitor element 1432







Printed in USA

SC27-3887-00



Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

Database Monitoring Guide and Reference

