



System Monitor Guide and Reference
Updated March, 2008



System Monitor Guide and Reference
Updated March, 2008

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 589.

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 www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at 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 International Business Machines Corporation 1993, 2008. All rights reserved.

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

Contents

About this book xi

Part 1. Monitoring database systems 1

Chapter 1. Database system monitor . . 3

Comparison of DB2 monitors 3
Database system monitor data organization 5
Counter status and visibility 7
System monitor output: the self-describing data stream 7
Database system monitor memory requirements . . 8
Indoubt Transaction Manager overview 11

Chapter 2. System monitor switches . . 15

Setting monitor switches from the CLP 17
Setting monitor switches from a client application 18
Monitor switches self-describing data stream . . . 19

Chapter 3. Snapshot monitor 21

Access to system monitor data: SYSMON authority 21
Capturing database system snapshots using snapshot administrative views and table functions . 22
Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure 24
Accessing database system snapshots using snapshot table functions in SQL queries (with file access) 26
Snapshot monitor SQL Administrative Views . . . 27
SQL access to database system snapshots 30
Capturing a database snapshot from the CLP . . . 31
Snapshot monitor CLP commands 31
Capturing a database snapshot from a client application 34
Snapshot monitor API request types 35
Snapshot monitor sample output 37
Subsection snapshots 39
Global snapshots on partitioned database systems 40
Snapshot monitor self-describing data stream . . . 41

Chapter 4. Event monitors 45

Event types 45
Collecting information about database system events 47
Creating an event monitor 49
 Creating a table event monitor 49
 Event monitor table management 52
 Creating a file event monitor 56
 Event monitor file management 58
 Write-to-table and file event monitor buffering . 59
 Creating a pipe event monitor 60
 Event monitor named pipe management . . . 60
 Creating an event monitor for partitioned databases 61
Event monitor sample output 63

 Formatting file or pipe event monitor output from a command line 70
 Event records and their corresponding applications 70
 Event monitor self-describing data stream . . . 71
 Transferring event monitor data between systems 73

Chapter 5. Activity Monitor overview 77

Monitoring scenarios 80
 Scenario: Identifying costly applications using snapshot administrative views 80
 Scenario: Monitoring buffer pool efficiency using administrative views 82
Setting up an activity monitor 83
Progress monitoring of the runtime rollback process 83
Using snapshot monitor data to monitor the reorganization of a partitioned table 84
Inactive statement tracking for DEADLOCK WITH DETAILS HISTORY event monitors 91

Chapter 6. Working with the Memory Visualizer 93

Memory Visualizer overview 95

Chapter 7. Monitoring database systems (Windows) 99

Introduction to Windows Management Instrumentation (WMI) 99
DB2 database system integration with Windows Management Instrumentation 99
Windows performance monitor introduction . . . 101
 Registering DB2 with the Windows performance monitor 101
 Enabling remote access to DB2 performance information 102
 Displaying DB2 database and DB2 Connect performance values 102
 Windows performance objects 102
 Accessing remote DB2 database performance information 103
 Resetting DB2 performance values 104

Part 2. System monitor elements 105

Chapter 8. Logical data groups. . . . 107

Snapshot monitor interface mappings to logical data groups 107
Snapshot monitor logical data groups and monitor elements 111
Event type mappings to logical data groups . . . 141
Event monitor logical data groups and monitor elements 144
Logical data groups affected by COLLECT ACTIVITY DATA settings 165

Chapter 9. Database system monitor elements 167

Server identification and status monitor elements	168
db2start_time - Start Database Manager Timestamp	168
server_instance_name - Server Instance Name	168
server_db2_type - Database Manager Type at Monitored (Server) Node	169
server_prdid - Server Product/Version ID	169
server_version - Server Version	170
service_level - Service Level	170
server_platform - Server Operating System	171
product_name - Product Name	171
db2_status - Status of DB2 Instance	171
time_zone_disp - Time Zone Displacement	172
Database identification and status monitor elements	172
db_name - Database Name	172
db_path - Database Path.	173
db_conn_time - Database Activation Timestamp	174
conn_time - Time of Database Connection	174
disconn_time - Database Deactivation Timestamp	174
db_status - Status of Database.	175
catalog_node_name - Catalog Node Network Name	175
db_location - Database Location	176
catalog_node - Catalog Node Number	176
last_backup - Last Backup Timestamp	176
db_storage_path - Automatic storage path.	177
num_db_storage_paths - Number of automatic storage paths	177
sto_path_free_sz - Automatic Storage Path Free Space	177
fs_used_size - Amount of Space Used on a File System	178
fs_total_size - Total Size of a File System	178
fs_id - Unique File System Identification Number	179
fs_type - File System Type	179
Application identification and status monitor elements	180
agent_id - Application Handle (agent ID)	180
appl_status - Application Status	181
codepage_id - ID of Code Page Used by Application	183
status_change_time - Application Status Change Time	183
appl_id_oldest_xact - Application with Oldest Transaction	184
smallest_log_avail_node - Node with Least Available Log Space	184
appl_name - Application Name	185
appl_id - Application ID.	185
sequence_no - Sequence number monitor element	187
auth_id - Authorization ID	188
session_auth_id - Session Authorization ID	189
client_prdid - Client Product/Version ID	189
client_db_alias - Database Alias Used by Application	190

host_prdid - Host Product/Version ID	190
is_system_appl - Is System Application monitor element	191
outbound_appl_id - Outbound Application ID	191
outbound_sequence_no - Outbound Sequence Number	192
execution_id - User Login ID	192
corr_token - DRDA Correlation Token	193
client_pid - Client Process ID	193
client_platform - Client Operating Platform	194
client_protocol - Client Communication Protocol	194
territory_code - Database Territory Code	195
appl_priority - Application Agent Priority	195
appl_priority_type - Application Priority Type	196
authority_lvl - User Authorization Level	196
authority_bitmap - User Authorization Level monitor element	197
node_number - Node Number	198
coord_node - Coordinating Node.	199
appl_con_time - Connection Request Start Timestamp	199
connections_top - Maximum Number of Concurrent Connections.	200
conn_complete_time - Connection Request Completion Timestamp	200
prev_uow_stop_time - Previous Unit of Work Completion Timestamp	201
uow_start_time - Unit of Work Start Timestamp	201
uow_stop_time - Unit of Work Stop Timestamp	202
uow_elapsed_time - Most Recent Unit of Work Elapsed Time	202
uow_comp_status - Unit of Work Completion Status	203
uow_status - Unit of Work Status.	203
appl_idle_time - Application Idle Time	204
DB2 agent information monitor elements	204
Database manager configuration monitor elements	205
Agents and connections monitor elements.	205
Memory pool monitor elements	216
Sort monitor elements	219
Hash join monitor elements	227
On-Line Analytical Processing (OLAP) monitor elements	230
Fast communications manager (FCM) monitor elements	232
Database configuration monitor elements	235
Buffer pool activity monitor elements	235
Non-buffered I/O activity monitor elements	268
Catalog cache monitor elements	272
Package cache monitor elements	275
SQL workspaces monitor elements	279
Database heap monitor elements	286
Logging monitor elements	286
Database and application activity monitor elements	296
blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element	296
Locks and deadlocks monitor elements.	297
Lock wait information monitor elements	312
Rollforward monitoring monitor elements.	318
Table space activity monitor elements	320
Table activity monitor elements	342

Table reorganization monitor elements	355	gw_total_cons - Total Number of Attempted Connections for DB2 Connect	431
SQL cursors monitor elements	360	gw_cur_cons - Current Number of Connections for DB2 Connect	432
SQL and XQuery statement activity monitor elements	363	gw_cons_wait_host - Number of Connections Waiting for the Host to Reply	432
SQL statement details monitor elements	373	gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request	432
Subsection details monitor elements	392	gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing	433
Dynamic SQL monitor elements	397	sql_stmts - Number of SQL Statements Attempted	433
Intra-query parallelism monitor elements	399	sql_chains - Number of SQL Chains Attempted	434
CPU usage monitor elements	400	open_cursors - Number of Open Cursors	435
Snapshot monitoring monitor elements	404	dcs_appl_status - DCS Application Status	435
Event monitoring monitor elements	406	agent_status - DCS Application Agents	436
Utilities monitor elements	412	host_ccsid - Host Coded Character Set ID	436
High availability disaster recovery (HADR) monitor elements	418	outbound_comm_protocol - Outbound Communication Protocol	436
hadr_role - HADR Role	418	outbound_comm_address - Outbound Communication Address	437
hadr_state - HADR State monitor element	419	inbound_comm_address - Inbound Communication Address	437
hadr_syncmode - HADR Synchronization Mode monitor element	420	inbound_bytes_received - Inbound Number of Bytes Received	437
hadr_connect_status - HADR Connection Status monitor element	420	outbound_bytes_sent - Outbound Number of Bytes Sent	438
hadr_connect_time - HADR Connection Time monitor element	421	outbound_bytes_received - Outbound Number of Bytes Received	438
hadr_heartbeat - HADR Heartbeat monitor element	422	inbound_bytes_sent - Inbound Number of Bytes Sent	439
hadr_local_host - HADR Local Host monitor element	422	outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent	439
hadr_local_service - HADR Local Service monitor element	423	outbound_bytes_received_top - Maximum Outbound Number of Bytes Received	440
hadr_remote_host - HADR Remote Host monitor element	423	outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent	440
hadr_remote_service - HADR Remote Service monitor element	424	outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received	440
hadr_remote_instance - HADR Remote Instance monitor element	424	max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes	441
hadr_timeout - HADR Timeout monitor element	425	max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes	441
hadr_primary_log_file - HADR Primary Log File monitor element	425	max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes	442
hadr_primary_log_page - HADR Primary Log Page monitor element	426	max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes	442
hadr_primary_log_lsn - HADR Primary Log LSN monitor element	426	max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes	442
hadr_standby_log_file - HADR Standby Log File monitor element	427	max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes	443
hadr_standby_log_page - HADR Standby Log Page monitor element	427	max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes	443
hadr_standby_log_lsn - HADR Standby Log LSN monitor element	428		
hadr_log_gap - HADR Log Gap	428		
hadr_peer_window - HADR peer window monitor element	428		
hadr_peer_window_end - HADR peer window end monitor element	429		
DB2 Connect monitor elements	430		
dcs_db_name - DCS Database Name	430		
host_db_name - Host Database Name	430		
gw_db_alias - Database Alias at the Gateway	430		
gw_con_time - DB2 Connect Gateway First Connect Initiated	431		
gw_connections_top - Maximum Number of Concurrent Connections to Host Database	431		

max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes	444
max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes	444
max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes	445
max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes	445
max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes	445
max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes	446
max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes	446
max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes	447
max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes.	447
max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes	448
max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element	448
max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes	448
max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element	449
max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes	449
max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes	450
max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms	450
max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms	451
max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms	451
max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms	452
max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms	452

max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms	452
network_time_top - Maximum Network Time for Statement	453
network_time_bottom - Minimum Network Time for Statement	453
xid - Transaction ID	454
elapsed_exec_time - Statement Execution Elapsed Time	454
host_response_time - Host Response Time.	455
num_transmissions - Number of Transmissions	455
num_transmissions_group - Number of Transmissions Group	456
con_response_time - Most Recent Response Time for Connect	456
con_elapsed_time - Most Recent Connection Elapsed Time	457
gw_comm_errors - Communication Errors.	457
gw_comm_error_time - Communication Error Time	458
blocking_cursor - Blocking Cursor	458
Transaction processor monitoring monitor elements	458
Federated database systems monitor elements	460
datasource_name - Data Source Name	460
disconnects - Disconnects	461
insert_sql_stmts - Inserts	461
update_sql_stmts - Updates	462
delete_sql_stmts - Deletes	462
create_nickname - Create Nicknames	463
passthru - Pass-Through	463
stored_procs - Stored Procedures	464
remote_locks - Remote Locks	464
sp_rows_selected - Rows Returned by Stored Procedures	465
select_time - Query Response Time	465
insert_time - Insert Response Time	466
update_time - Update Response Time	466
delete_time - Delete Response Time	467
create_nickname_time - Create Nickname Response Time	468
passthru_time - Pass-Through Time	468
stored_proc_time - Stored Procedure Time.	469
remote_lock_time - Remote Lock Time	469
Workload management monitor elements	470
activate_timestamp - Activate timestamp monitor element	470
activity_collected - Activity collected monitor element	470
activity_id - Activity ID monitor element	470
activity_secondary_id - Activity secondary ID monitor element	471
activity_type - Activity type monitor element	472
act_exec_time - Activity execution time monitor element	472
act_total - Activities total monitor element.	473
arm_correlator - Application response measurement correlator monitor element	473
bin_id - Histogram bin identifier monitor element	473

bottom - Histogram bin bottom monitor element	474	rows_returned_top - Actual rows returned top monitor element	489
concurrent_act_top - Concurrent activity top monitor element	474	sc_work_action_set_id - Service class work action set ID monitor element	489
concurrent_connection_top - Concurrent connection top monitor element	475	sc_work_class_id - Service class work class ID monitor element	490
concurrent_wlo_act_top - Concurrent WLO activity top monitor element	475	section_env - Section environment monitor element	490
concurrent_wlo_top - Concurrent workload occurrences top monitor element	475	service_class_id - Service class ID monitor element	490
coord_act_aborted_total - Coordinator activities aborted total monitor element	476	service_subclass_name - Service subclass name monitor element	491
coord_act_completed_total - Coordinator activities completed total monitor element	476	service_superclass_name - Service superclass name monitor element	491
coord_act_lifetime_top - Coordinator activity lifetime top monitor element	477	statistics_timestamp - Statistics timestamp monitor element	492
coord_act_rejected_total - Coordinator activities rejected total monitor element	477	temp_tablespace_top - Temporary table space top monitor element	492
coord_partition_num - Coordinator partition number monitor element	478	threshold_action - Threshold action monitor element	493
cost_estimate_top - Cost estimate top monitor element	478	threshold_domain - Threshold domain monitor element	493
coord_act_lifetime_avg - Coordinator activity lifetime average monitor element	478	threshold_maxvalue - Threshold maximum value monitor element	494
coord_act_queue_time_avg - Coordinator activity queue time average monitor element	479	threshold_name - Threshold name monitor element	494
coord_act_exec_time_avg - Coordinator activities execution time average monitor element	480	threshold_predicate - Threshold predicate monitor element	494
request_exec_time_avg - Request execution time average monitor element	480	threshold_queuesize - Threshold queue size monitor element	495
coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element	481	thresholdid - Threshold ID monitor element	495
coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element	482	time_completed - Time completed monitor element	496
db_work_action_set_id - Database work action set ID monitor element	482	time_created - Time created monitor element	496
db_work_class_id - Database work class ID monitor element	483	time_of_violation - Time of violation monitor element	496
histogram_type - Histogram type monitor element	483	time_started - Time started monitor element	497
last_wlm_reset - Time of last reset monitor element	484	top - Histogram bin top monitor element	497
num_threshold_violations - Number of threshold violations monitor element	484	uow_id - Unit of work ID monitor element	497
number_in_bin - Number in bin monitor element	485	wlo_completed_total - Workload occurrences completed total monitor element	498
parent_activity_id - Parent activity ID monitor element	485	work_action_set_id - Work action set ID monitor element	498
parent_uow_id - Parent unit of work ID monitor element	486	work_action_set_name - Work action set name monitor element	499
prep_time - Preparation time monitor element	486	work_class_id - Work class ID monitor element	499
queue_assignments_total - Queue assignments total monitor element	486	work_class_name - Work class name monitor element	499
queue_size_top - Queue size top monitor element	487	workload_id - Workload ID monitor element	500
queue_time_total - Queue time total monitor element	487	workload_name - Workload name monitor element	500
rows_fetched - Rows fetched monitor element	487	workload_occurrence_id - Workload occurrence identifier monitor element	501
rows_modified - Rows modified monitor element	488	Real-time statistics monitor elements	501
rows_returned - Rows returned monitor element	488	stats_cache_size - Size of statistics cache monitor element	501
		stats_fabrications - Total number of statistics fabrications monitor elements	502
		sync_runstats - Total number of synchronous RUNSTATS activities monitor element	502

async_runstats – Total number of asynchronous RUNSTATS requests monitor element	503
stats_fabricate_time – Total time spent on statistics fabrication activities monitor element	504
sync_runstats_time – Total time spent on synchronous RUNSTATS activities monitor element	504

Chapter 10. Database system monitor interfaces 507

Part 3. Monitoring database health 509

Chapter 11. Introduction to the health monitor 511

Health indicators	511
Health indicator process cycle	513
Enabling health alert notification	514

Chapter 12. Health Center overview 517

Investigating alert conditions	519
--	-----

Chapter 13. Health monitor 521

Health indicator data	522
Capturing database health snapshots	522
Capturing a database health snapshot using SQL table functions	522
Capturing a database health snapshot using the CLP	523
Capturing a database health snapshot from a client application	524
Health monitor sample output	526
Global health snapshots	528
Graphical tools for the health monitor	529
Retrieving health recommendations	531
Health recommendation queries with SQL.	531
Retrieving health recommendations using the CLP	531
Retrieving health recommendations using a client application	535
Resolving health monitor alerts using the Health Center.	536
Health indicator configuration.	537
Retrieving health indicator configuration using the CLP	539
Health indicator configuration updates using the CLP	539
Resetting health indicator configuration using the CLP	540
Configuring health indicators using a client application	541
Configuring health indicators using the Health Center.	543
Health monitor alert actions on combined states	544

Part 4. Health indicators 547

Chapter 14. Health monitor interface mappings to logical data groups . . . 549

Chapter 15. Health indicators summary 551

Health indicator format	553
Table space storage health indicators	554
Health indicators for DMS table spaces.	554
db.auto_storage_util - Database automatic storage utilization health indicator	554
ts.ts_auto_resize_status - Table space automatic resize status health indicator	555
ts.ts_util_auto_resize - Automatic resize table space utilization health indicator	556
ts.ts_util - Table Space Utilization.	556
tsc.tscont_util - Table Space Container Utilization	557
ts.ts_op_status - Table Space Operational State	558
tsc.tscont_op_status - Table Space Container Operational State	558
Sorting health indicators.	558
db2.sort_privmem_util - Private Sort Memory Utilization	558
db.sort_shrmem_util - Shared Sort Memory Utilization	559
db.spilled_sorts - Percentage of Sorts That Overflowed	560
db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization.	560
Database manager (DBMS) health indicators	561
db2.db2_op_status - Instance Operational State	561
Instance Highest Severity Alert State	561
Database health indicators	562
db.db_op_status - Database Operational State	562
Database Highest Severity Alert State	562
Maintenance health indicators.	563
db.tb_reorg_req - Reorganization Required	563
db.tb_runstats_req - Statistics Collection Required	563
db.db_backup_req - Database Backup Required	564
High availability disaster recovery (HADR) health indicators.	564
db.hadr_op_status - HADR Operational Status	564
db.hadr_delay - HADR Log Delay	565
Logging health indicators	565
db.log_util - Log Utilization	565
db.log_fs_util - Log File System Utilization	565
Application concurrency health indicators.	566
db.deadlock_rate - Deadlock Rate	566
db.locklist_util - Lock List Utilization	567
db.lock_escal_rate - Lock Escalation Rate	567
db.apps_waiting_locks - Percentage of Applications Waiting on Locks	568
Package cache, catalog cache, and workspace health indicators	569
db.catcache_hitratio - Catalog Cache Hit Ratio	569
db.pkgcache_hitratio - Package Cache Hit Ratio	569
db.shrworkspace_hitratio - Shared Workspace Hit Ratio	569
Memory health indicators	570

db2.mon_heap_util - Monitor Heap Utilization	570
db.db_heap_util - Database Heap Utilization	570
Federated health indicators	571
db.fed_nicknames_op_status - Nickname Status	571
db.fed_servers_op_status - Data Source Server Status	571

Chapter 16. Health monitor interfaces 573

Health monitor SQL table functions	574
Health monitor CLP commands	574
Health monitor API request types	575

Part 5. Appendixes 577

Appendix A. Overview of the DB2 technical information 579

DB2 technical library in hardcopy or PDF format	579
---	-----

Ordering printed DB2 books	582
Displaying SQL state help from the command line processor	582
Accessing different versions of the DB2 Information Center	583
Displaying topics in your preferred language in the DB2 Information Center	583
Updating the DB2 Information Center installed on your computer or intranet server	584
DB2 tutorials	585
DB2 troubleshooting information	586
Terms and Conditions	586

Appendix B. Notices 589

Index 593

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. Monitoring database systems

Chapter 1. Database system monitor

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 do the following, 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 instance, altering database manager configuration parameters, adding indexes, or modifying SQL queries).

There are two primary tools with which you can access system monitor information, each serving a different purpose: the snapshot monitor and event monitors. The snapshot monitor enables you to capture a picture of the state of database activity at a particular point in time (the moment the snapshot is taken). Event monitors log data as specified database events occur.

The system monitor provides multiple means of presenting monitor data to you. For both snapshot and event monitors you have the option of storing monitor information in files or SQL tables, viewing it on screen (directing it to standard-out), or processing it with a client application.

Comparison of DB2 monitors

DB2 Version 9.5 provides you with several ways to monitor your database system. The snapshot monitor, event monitor, and health monitor each fill different monitoring needs. The following table provides a brief overview and contrasts the characteristics of the different monitors.

Table 1. Comparison of DB2 Version 9.5 Monitors

	Snapshot monitor	Event monitor	Health monitor
Description	<ul style="list-style-type: none">• Real-time monitoring.• Provides a picture of the state of the database at the current point in time.• Data returned can be used to check database status, identify potential problem areas. When captured at regular intervals, can reveal trends in database activity.	<ul style="list-style-type: none">• Real-time, trigger-based monitoring.• Records state of the database whenever a specific type of event occurs, describing database activity over a period of time.• Provides detailed data for pinpointing/ diagnosing problems	<ul style="list-style-type: none">• Exception-based monitoring.• Flags abnormal or potentially problematic conditions in the database.• Provides high-level picture of database health. Points out general areas of concern for further investigation.

Table 1. Comparison of DB2 Version 9.5 Monitors (continued)

	Snapshot monitor	Event monitor	Health monitor
Level of data collected	<ul style="list-style-type: none"> • Database manager • Database • Application (includes statement level information) • Buffer pool • Table Space • Table • Lock and Lockwait • Dynamic SQL • DCS Application and Database 	<ul style="list-style-type: none"> • Database • Connection (equivalent to Snapshot application level) • Buffer pool • Table Space • Table • Deadlock • Transaction • Statement 	<ul style="list-style-type: none"> • Database manager • Database • Table space • Table space container
Activated/enabled	Set specific monitor switches to ON ¹ ; <code>TIMESTAMP=ON</code> by default. Switches can be enabled per application (using the <code>update monitor switchescommand</code>) or at the database manager level (using the <code>update dbm cfg command</code>).	Create event monitor using the <code>AUTOSTART</code> option, or set event monitor to state ²	Enabled by default. To deactivate, set <code>health_mon</code> database manager configuration parameter to OFF
When data is collected	User issues snapshot table functions, snapshot APIs, <code>SNAP_WRITE_FILE</code> stored procedure or <code>get snapshot</code> command from the CLP, or issues <code>SELECT</code> from snapshot administrative views	Specified event occurs ³	Collected by default at preset intervals

Table 1. Comparison of DB2 Version 9.5 Monitors (continued)

	Snapshot monitor	Event monitor	Health monitor
Means of retrieving/analyzing data	Use snapshot table functions, snapshot administrative views, CLP, snapshot monitor API, or Activity monitor graphical tool.	<ul style="list-style-type: none"> For table event monitors, access event tables with SQL or use Event Analyzer graphical tool For named pipe event monitors, use db2evmon utility or a client program that reads the monitor data from the pipe. For file event monitors, use the Event Analyzer, the db2evmon utility, or a client program that reads the monitor data from the file 	<ul style="list-style-type: none"> Receive e-mail or page notifications about alerts Retrieve health data using SQL table functions, snapshot from CLP, health snapshot API Use Health Center to view current alerts Address alerts by using recommendations advisor graphical tool, returning recommendations from CLP, stored procedure, or API
Overhead	Varies with number of switches enabled and the type of workload run on the instance; may increase system workload by 3-10%	Varies with type of data being monitored (for example, the statement event monitor returns detailed data for each executed statement) and how selective event monitor is (for example, whether it uses a WHERE clause)	Minimal overhead for Health monitoring. Additional overhead incurred for graphical tools invoked from Health Center.

Note:

1. Some monitor information is not under switch control but instead is collected all the time. Other types of monitor information are only collected when specific switches are turned on.
2. A detailed deadlock event monitor, DB2DETAILDEADLOCK, is created by default for each database and starts when the database is activated.
3. You can flush and event monitor buffer to force an event monitor to write out its current data.

Database system monitor data organization

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 are the available element types in which monitor elements 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 database manager 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 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 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.

Database system monitor memory requirements

The memory required for maintaining database system monitor data is allocated from the monitor heap. Monitor heap size is controlled by the **mon_heap_sz** configuration parameter. The amount of memory required for monitoring activity varies widely, depending on 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 **mon_heap_sz** configuration 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:

$$\begin{array}{r} \text{(Storage used by applications} \\ \text{Storage used by event monitors} \\ \text{Storage used by monitoring applications} \\ \text{Storage used by Gateway applications)} \end{array} \begin{array}{r} + \\ + \\ + \\ \end{array} / 4096$$

Storage 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, applname, workstation name and accounting string.

Storage used by each event monitor

- 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 “Storage used by each application.”
- 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.
- If an event monitor is of type ACTIVITIES:
 - 2 * BUFFERSIZE are not allocated. Instead, the total amount of memory used by queued activity event monitor records is controlled by the DB2_EVMON_EVENT_LIST_SIZE registry variable.
 - Each event_activity event monitor record takes approximately 4900 bytes.

- Each event_activitystmt event monitor record takes approximately 2500 bytes + the size of the statement text.
- Each event_activityvals event monitor record takes approximately 900 bytes.

Storage 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

Storage 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

Indoubt Transaction Manager overview

Use the Indoubt Transaction Manager window to work with indoubt transactions. The window lists all indoubt transactions for a selected database and one or more selected partitions.

An indoubt transaction is a global transaction that was left in an indoubt state. DB2 provides heuristic actions that database administrators can perform on indoubt transactions when the resource owner, such as the database administrator, cannot wait for the Transaction Manager to perform the resync action. This condition may occur if, for example, the communication line is broken, and an indoubt transaction is tying up needed resources such as locks on tables and indexes, log space, and storage used by the transaction.

While it is preferable for the Transaction Manager to initiate the re-sync action, there may be times when you may have to perform the heuristic actions on the indoubt transactions. In these cases, use the heuristic actions with caution and only as a last resort and follow these guidelines.

- The *gtrid* portion of the transaction ID is the global transaction ID that is identical to that in other resource managers (RM) that participate in the global transaction.
- Use your knowledge of the application and the operating environment to identify the other participating resource managers,
- If the transaction manager is CICS®, and the only resource manager is a CICS resource, perform a heuristic rollback.
- If the transaction manager is not CICS, use it to determine the status of the transaction that has the same *gtrid* as the indoubt transaction.
- If, at least, one resource manager has committed or rolled back, perform a heuristic commit or rollback.
- If all the transactions are in the prepared state, perform a heuristic rollback.
- If, at least, one of the resource managers is not available, perform a heuristic rollback.

To open the Indoubt Transaction Manager on Intel® platforms, from the **Start** menu, click **Start -> Programs -> IBM DB2 -> Monitoring Tools -> Indoubt Transaction Manager**.

To open the Indoubt Transaction Manager using the command line in UNIX® or on Intel, run the following command:

```
db2indbt
```

You can perform the following heuristic actions on indoubt transactions:

- Forget

This permits the resource manager to erase knowledge of a heuristically completed transaction by removing the log records and releasing log pages. A heuristically completed transaction is one that has been committed or rolled back heuristically. You can use the forget action on transactions that are

heuristically committed or rolled back for a selected database and one or more selected partitions. To forget an indoubt transaction, select a database and partition and then right-click a transaction with a status of **Committed** or **Rolled back** and select **Forget** from the pop-up menu. A confirmation message displays.

- **Commit**

This commits an indoubt transaction that is prepared to be committed. If the operation succeeds, the transaction's state becomes heuristically committed. To commit an indoubt transaction, select a database and partition and then right-click a transaction with a status of **Indoubt** or **Missing commit acknowledgement** and select **Commit** from the pop-up menu. A confirmation message displays.

- **Rollback**

This rolls back an indoubt transaction that has been prepared. If the operation succeeds, the transaction's state becomes heuristically rolled back. To roll back an indoubt transaction, select a database and partition and then right-click a transaction with a status of **Indoubt** or **Ended** and select **Rollback** from the pop-up menu. A confirmation message displays.

To perform these actions on indoubt transactions you must have SYSADM or DBADM authority.

The columns in the Indoubt Transaction Manager window provide named views that you can use to organize and display indoubt transactions in different ways. The following list describes each of the columns in the interface:

Status

The indoubt status of the transaction, namely Committed (c), Ended (e), Indoubt (i), Missing commit acknowledgement (m), and Rolled back (r):

Committed

Transactions in this state have been heuristically committed.

Ended

Transactions in this state may have timed out.

Indoubt

Transactions in this state are waiting to be committed or rolled back.

Missing commit acknowledgement

The Transaction Manager is waiting to receive an acknowledgement before committing the transaction.

Rolled back

Transactions in this state have been heuristically rolled back

Timestamp

The time stamp on the server when the transaction entered the prepared (indoubt) state. The time is the local time to the client.

Transaction ID

The XA identifier assigned by the transaction manager to uniquely identify a global transaction.

Application ID

The application identifier assigned by the database manager for this transaction.

Authorization ID

The user ID of the user who ran the transaction.

Sequence Number

The sequence number assigned by the database manager as an extension to the application identifier.

Partition

The partition on which the indoubt transaction exists.

Originator

Indicates whether the transaction was originated by XA or by DB2 in a partitioned database environment.

Log Full

Indicates whether this transaction caused a log full condition.

Type The type information that shows the role of the database in each indoubt transaction.

- **TM** indicates the indoubt transaction is using the database as a transaction manager database.
- **RM** indicates the indoubt transaction is using the database as a resource manager. This means that it is one of the databases participating in the transaction, but is not the transaction manager database.

Chapter 2. System monitor switches

Collecting system monitor data introduces processing overhead 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. Another form of overhead incurred by the system monitor is increased 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 overhead 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 below. 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 below. 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 2. 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 above 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 monitor switches from the CLP

Monitor switches control the collection of data by the database manager. By setting certain monitor switches to ON, you can collect specific types of monitor data.

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.

- 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 above 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 above 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
```

Now that you have set the desired monitor switches and confirmed the switch settings, you are ready to capture and collect monitor data.

Setting monitor switches from a client application

Monitor switches control the collection of data by the database manager. By setting certain monitor switches to ON, you can collect specific types of monitor data.

The application performing any monitor switch updates must have an instance attachment. You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to use the db2MonitorSwitches API.

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);
```

Now that you have set the desired monitor switches and confirmed the switch settings, you are ready to capture and collect monitor data.

Monitor switches self-describing data stream

After you update or view the current monitor switch settings with the `db2MonitorSwitches` API, the API returns the switch settings as a self-describing data stream. Figure 1 on page 20 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.

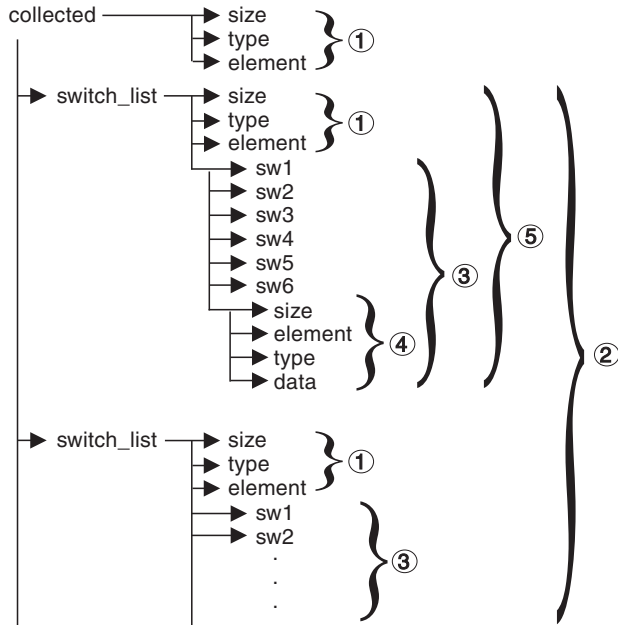


Figure 1. Switch List Monitor Data Stream

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.

Chapter 3. 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. To obtain monitor information for all database activity during a given period use an event monitor.

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.

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 replaces the `DB2_SNAPSHOT_NOAUTH` registry variable as 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
 - 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 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.

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.

While new snapshot table functions may 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 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
```


Snapshot administrative views and table functions cannot be used in conjunction with either of the following:

- Monitor switches commands/APIs
- Monitor reset commands/APIs

This restriction includes:

- GET MONITOR SWITCHES
- UPDATE MONITOR SWITCHES
- RESET MONITOR

This limitation is due to the fact that such commands use an INSTANCE ATTACH, while snapshot table functions make use of DATABASE CONNECTs.

To capture a snapshot using a snapshot administrative view:

1. To capture a snapshot using a snapshot administrative view:
 - a. 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.
 - b. 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.
 - c. 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
```

2. To capture a snapshot using a snapshot table function:
 - a. 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.
 - b. Determine the type of snapshot you need to capture.
 - c. 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:

- 1) 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_V95
 - SNAP_GET_DB_MEMORY_POOL
 - SNAP_GET_DETAILLOG_V91
 - SNAP_GET_HADR
 - SNAP_GET_STORAGE_PATHS
 - SNAP_GET_APPL_V95
 - SNAP_GET_APPL_INFO_V95
 - SNAP_GET_AGENT
 - SNAP_GET_AGENT_MEMORY_POOL
 - SNAP_GET_STMT
 - SNAP_GET_SUBSECTION
 - SNAP_GET_BP_V95
 - SNAP_GET_BP_PART
- 2) 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, SYSMAINT, 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.)

You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to capture a database snapshot with the SNAP_WRITE_FILE stored procedure.

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 3. Snapshot request types

Snapshot table function	Snapshot request type
SNAP_GET_AGENT	APPL_ALL
SNAP_GET_AGENT_MEMORY_POOL	APPL_ALL
SNAP_GET_APPL_V95	APPL_ALL

Table 3. Snapshot request types (continued)

Snapshot table function	Snapshot request type
SNAP_GET_APPL_INFO_V95	APPL_ALL
SNAP_GET_STMT	APPL_ALL
SNAP_GET_SUBSECTION	APPL_ALL
SNAP_GET_BP_PART	BUFFERPOOLS_ALL
SNAP_GET_BP_V95	BUFFERPOOLS_ALL
SNAP_GET_DB_V95	DBASE_ALL
SNAP_GET_DETAILLOG_V91	DBASE_ALL
SNAP_GET_DB_MEMORY_POOL	DBASE_ALL
SNAP_GET_HADR	DBASE_ALL
SNAP_GET_STORAGE_PATHS	DBASE_ALL
SNAP_GET_DBM_V95	DB2
SNAP_GET_DBM_MEMORY_POOL	DB2
SNAP_GET_FCM	DB2
SNAP_GET_FCM_PART	DB2
SNAP_GET_SWITCHES	DB2
SNAP_GET_DYN_SQL_V95	DYNAMIC_SQL
SNAP_GET_LOCK	DBASE_LOCKS
SNAP_GET_LOCKWAIT	APPL_ALL
SNAP_GET_TAB_V91	DBASE_TABLES
SNAP_GET_TAB_REORG	DBASE_TABLES
SNAP_GET_TBSP_V91	DBASE_TABLESPACES
SNAP_GET_TBSP_PART_V91	DBASE_TABLESPACES
SNAP_GET_CONTAINER_V91	DBASE_TABLESPACES
SNAP_GET_TBSP_QUIESCER	DBASE_TABLESPACES
SNAP_GET_TBSP_RANGE	DBASE_TABLESPACES
SNAP_GET_UTIL	DB2
SNAP_GET_UTIL_PROGRESS	DB2

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 3 on page 24, 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 wish 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.

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, SYSMAINT, 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.

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, SYSMAINT, or SYSMON authority.

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.

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_V91 (CAST(NULL AS VARCHAR(1)),
                                       CAST(NULL AS INTEGER))) AS SNAP_GET_TBSP_V91
```

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_V95(CAST(NULL AS VARCHAR(1)),
                     CAST(NULL AS INTEGER)))
as SNAP_GET_APPL_V95
```

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.SNAPBP SQL administrative view captures a snapshot of buffer pool information. The following table lists each available snapshot monitor administrative view.

Table 4. 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.

Table 4. Snapshot Monitor SQL Administrative Views (continued)

Monitor level	SQL Administrative Views	Information returned
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.
Database	SYSIBMADM.SNAPHADR	Database level information about high availability disaster recovery.
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.
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 4. Snapshot Monitor SQL Administrative Views (continued)

Monitor level	SQL Administrative Views	Information returned
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.
Database	SYSIBMADM.SNAPDETAILLOG	Database level information about log files.
Database	SYSIBMADM.SNAPSTORAGE_PATHS	Returns a list of automatic storage paths for the database including 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, 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 5. 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.

Table 5. Snapshot Monitor SQL Administrative Convenience Views (continued)

SQL Administrative Convenience Views	Information returned
ISYSIBMADM.LOCKWAIT	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
- file access

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, SYSMOINT, or SYSMON authority.

To capture snapshot information using direct access:

1. Optional: Set and check the status of the monitor switches.
2. Capture database system snapshots using SQL.

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 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`, `SYSMAINT`, or `SYSMON` authority.

The following tasks are performed by the `SYSADM`, `SYSCTRL`, `SYSMAINT`, or `SYSMON` user who captures database system snapshot information to a file.

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.
3. Capture database system snapshot information to a file .

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.

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.

1. Optional: Set and check the status of the monitor switches.
2. From the CLP, issue the `GET SNAPSHOT` command with the desired parameters. In the following example, a snapshot captures database manager level information:

```
db2 get snapshot for dbm
```

3. 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
```

4. 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 6. 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).
Application	<code>get snapshot for applications on <i>dbname</i></code>	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	<code>get snapshot for all applications</code>	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 6. Snapshot Monitor CLP Commands (continued)

Monitor level	CLP command	Information returned
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 was accessed 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 on 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.
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()`.

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.

1. Optional: Set and check the status of the monitor switches.
2. Include the following DB2 libraries: `sqlmon.h` and `db2ApiDf.h`. These are found in the include subdirectory under `sqllib`.

```
#include <db2ApiDf.h>
#include <sqlmon.h>
```

3. Set snapshot buffer unit size to 100 KB.
4. Declare the `sqlca`, `sqlma`, `db2GetSnapshotData`, and `sqlm_collected` structures. Also, initialize a pointer to contain the snapshot buffer, and establish the buffer's size.

```
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;
```

5. Initialize the `sqlma` structure and specify that the snapshot to be captured is of database manager level information.

```
pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
memset(pRequestedDataGroups, '\0', SQLMASIZE(1));
pRequestedDataGroups->obj_num = 1;
pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
```

6. Initialize the buffer which is to hold the snapshot output.

```
snapshotBuffer = (char *)malloc(snapshotBufferSize);
memset (snapshotBuffer, '\0', snapshotBufferSize);
```

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_DEFAULT;
```

8. 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);

```
9. 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);
}

```
  10. Process the snapshot monitor data stream.
  11. 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 7. 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.                                                                                                                                                                                                                            |
| 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. |

Table 7. Snapshot Monitor API Request Types (continued)

| Monitor level | API request type       | Information returned                                                                                                                                                                                                                   |
|---------------|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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).                        |
| 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). |

Table 7. Snapshot Monitor API Request Types (continued)

| Monitor level | API request type          | Information returned                                                                                                                                                                                                                                                                    |
|---------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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.                                                                                                                                                                                                      |

## Snapshot monitor sample output

To illustrate the nature of the snapshot monitor, here is an example of a snapshot being taken using the CLP, along with its corresponding output. The objective in this example is to obtain a list of the locks held by applications connected to the SAMPLE database. The steps taken are as follows:

1. Connect to the sample database:  
db2 connect to sample
2. Turn on the LOCK switch with the UPDATE MONITOR SWITCHES command, so that the time spent waiting for locks is collected:  
db2 update monitor switches using LOCK on
3. Issue a command or statement that will require locks on the database catalogs. In this case, we will declare, open, and fetch a cursor:



```

db2 -c- declare c1 cursor for
 select * from staff where job='Sales' for update
db2 -c- open c1
db2 -c- fetch c1

```

4. Take the database lock snapshot, using the GET SNAPSHOT command:

```
db2 get snapshot for locks on sample
```

After the GET SNAPSHOT command is issued from the CLP, the snapshot output is directed to the screen.

Database Lock Snapshot

```

Database name = SAMPLE
Database path = C:\DB2\NODE0000\SQL00001\
Input database alias = SAMPLE
Locks held = 5
Applications currently connected = 1
Agents currently waiting on locks = 0
Snapshot timestamp = 06-05-2002 17:08:25.048027

```

```

Application handle = 8
Application ID = *LOCAL.DB2.0098C5210749
Sequence number = 0001
Application name = db2bp.exe
CONNECT Authorization ID = DB2ADMIN
Application status = UOW Waiting
Status change time = Not Collected
Application code page = 1252
Locks held = 5
Total wait time (ms) = 0

```

List Of Locks

```

Lock Name = 0x02000300050000000000000052
Lock Attributes = 0x00000000
Release Flags = 0x00000001
Lock Count = 1
Hold Count = 0
Lock Object Name = 5
Object Type = Row
Tablespace Name = USERSPACE1
Table Schema = DB2ADMIN
Table Name = STAFF
Mode = U

```

```

Lock Name = 0x02000300000000000000000054
Lock Attributes = 0x00000000
Release Flags = 0x00000001
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 = 0x41414141414A48520000000041
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 inter-partition 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` or `dynexpln` commands. 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 run-time information with the access plan, which is compile-time information. With inter-partition 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 tablequeue 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 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 `apl_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 2 shows the structure of the data stream and Table 8 on page 42 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.

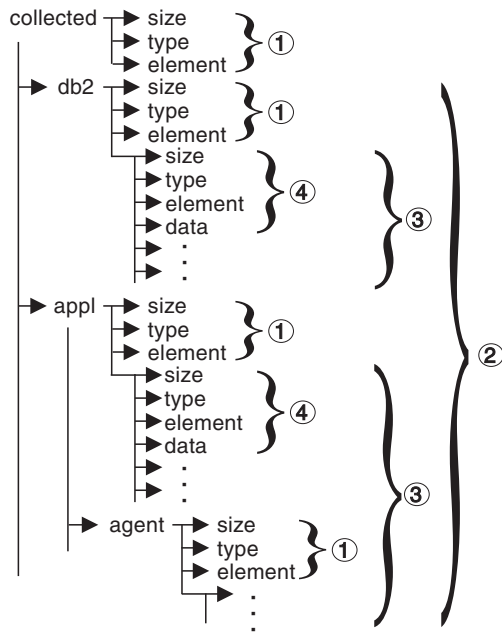


Figure 2. Snapshot Monitor Data Stream

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 8. Sample Snapshot Data Stream

| Logical Data Group | Data Stream         | Description                                      |
|--------------------|---------------------|--------------------------------------------------|
| collected          | 1000                | Size of snapshot data (in bytes).                |
|                    | header              | Indicates the start of a logical data group.     |
|                    | collected           | Name of the logical data group.                  |
|                    | 4                   | Size of the data stored in this monitor element. |
|                    | u32bit              | Monitor element type - unsigned 32 bit numeric.  |
|                    | server_db2_type     | The name of the monitor element collected.       |
|                    | sqlf_nt_server      | The collected value for this element.            |
|                    | 2                   | Size of the data stored in this monitor element. |
|                    | u16bit              | Monitor element type - unsigned 16 bit numeric.  |
|                    | node_number         | The name of the monitor element collected.       |
|                    | 3                   | The collected value for this element.            |
|                    | db2                 | 200                                              |
| header             |                     | Indicates the start of a logical data group.     |
| db2                |                     | Name of the logical data group.                  |
|                    | 4                   | Size of the data stored in this monitor element. |
|                    | u32bit              | Monitor element type - unsigned 32 bit numeric.  |
|                    | sort_heap_allocated | The name of the monitor element collected.       |
|                    | 16                  | The collected value for this element.            |
|                    | 4                   | Size of the data stored in this monitor element. |
|                    | u32bit              | Monitor element type - unsigned 32 bit numeric.  |
|                    | local_cons          | The name of the monitor element collected.       |
|                    | 3                   | The collected value for this element.            |
|                    | ...                 | ...                                              |
| appl               | 100                 | Size of the appl element data in the snapshot.   |
|                    | header              | Indicates the start of a logical data group.     |
|                    | appl                | Name of the logical data group.                  |
|                    | 4                   | Size of the data stored in this monitor element. |
|                    | u32bit              | Monitor element type - unsigned 32 bit numeric.  |
|                    | locks_held          | The name of the monitor element collected.       |
|                    | 3                   | The collected value for this element.            |
|                    | ...                 | ...                                              |

Table 8. 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.                  |
|                    | 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:

$$\text{size of the db2 logical data grouping} + \text{sizeof(sqlm\_header\_info)}$$



---

## Chapter 4. Event monitors

Event monitors are used to collect information about the database and any connected applications when specified events occur. Events represent transitions in database activity such as connections, deadlocks, statements, or transactions. You can define an event monitor by the type of event or events you want it to monitor. For example, a deadlock event monitor waits for a deadlock to occur; when one does, it collects information about the applications involved and the locks in contention.

By default, all databases have an event monitor defined named DB2DETAILDEADLOCK, which records detailed information about deadlock events. The DB2DETAILDEADLOCK event monitor starts automatically when the database starts.

Whereas the snapshot monitor is typically used for preventative maintenance and problem analysis, event monitors are used to alert administrators to immediate problems or to track impending ones.

To create an event monitor, use the CREATE EVENT MONITOR SQL statement. Event monitors collect event data only when they are active. To activate or deactivate an event monitor, use the SET EVENT MONITOR STATE SQL statement. The status of an event monitor (whether it is active or inactive) can be determined by the SQL function EVENT\_MON\_STATE.

When the CREATE EVENT MONITOR SQL statement is executed, the definition of the event monitor it creates is stored in the following database system catalog tables:

- SYSCAT.EVENTMONITORS: event monitors defined for the database.
- SYSCAT.EVENTS: events monitored for the database.
- SYSCAT.EVENTTABLES: target tables for table event monitors.

Each event monitor has its own private logical view of the instance's data in the monitor elements. If a particular event monitor is deactivated and then reactivated, its view of these counters is reset. Only the newly activated event monitor is affected; all other event monitors will continue to use their view of the counter values (plus any new additions).

Event monitor output can be directed to non-partitioned SQL tables, a file, or a named pipe.

---

### Event types

Event monitors return information for the event types specified in the CREATE EVENT MONITOR statement. For each event type, monitoring information is collected at a certain point in time.

The following table lists available event types, when the monitoring data is collected, and the information available for each event type. The available event types in the first column correspond to the keywords used in the CREATE EVENT MONITOR statement, where the event type is defined.

In addition to the defined events where data 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.

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.

Table 9. Event Types

| Event type                            | When data is collected  | Available information                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEADLOCKS                             | Detection of a deadlock | Applications involved, and locks in contention.                                                                                                                                                                                                                                                                                                                                                         |
| DEADLOCKS WITH DETAILS                | Detection of a deadlock | 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.                                          |
| DEADLOCKS WITH DETAILS HISTORY        | Detection of a deadlock | 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. |
| DEADLOCKS WITH DETAILS HISTORY VALUES | Detection of a deadlock | 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.                                                                       |
| STATEMENTS                            | End of SQL statement    | Statement start or stop time, CPU used, text of dynamic SQL, SQLCA (return code of SQL statement), and other metrics such as fetch count.<br><b>Note:</b> Statement start or stop time is unavailable when the Timestamp switch is off.                                                                                                                                                                 |
|                                       | End of subsection       | For partitioned databases: CPU consumed, execution time, table and tablequeue information.                                                                                                                                                                                                                                                                                                              |
| TRANSACTIONS                          | End of unit of work     | UOW work start or stop time, previous UOW time, CPU consumed, locking and logging metrics. Transaction records are not generated if running with XA.                                                                                                                                                                                                                                                    |
| CONNECTIONS                           | End of connection       | All application level counters.                                                                                                                                                                                                                                                                                                                                                                         |



Table 9. Event Types (continued)

| Event type           | When data is collected                                                                                                                                                                                                                                                                                                                                                                                  | Available information                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DATABASE             | Database deactivation                                                                                                                                                                                                                                                                                                                                                                                   | All database level counters.                                                                                                                                                                                                                                                                              |
| BUFFERPOOLS          | Database deactivation                                                                                                                                                                                                                                                                                                                                                                                   | Counters for buffer pool, prefetchers, page cleaners and direct I/O for each buffer pool.                                                                                                                                                                                                                 |
| TABLESPACES          | Database deactivation                                                                                                                                                                                                                                                                                                                                                                                   | Counters for buffer pool, prefetchers, page cleaners and direct I/O for each table space.                                                                                                                                                                                                                 |
| TABLES               | Database deactivation                                                                                                                                                                                                                                                                                                                                                                                   | Rows read or written for each table.                                                                                                                                                                                                                                                                      |
| Activities           | <p>Completion of an activity that executed in a service class, workload or work class that had its COLLECT ACTIVITY DATA option turned on. Data is also collected for the targeted activity at the instant the WLM_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure is executed.</p> <p>Data is also collected if the activity violates a threshold that has the COLLECT ACTIVITY DATA option enabled.</p> | Activity level data. If WITH DETAILS was specified as part of COLLECT ACTIVITY DATA, this will include statement and compilation environment information for those activities that have it. If AND VALUES was also specified, this will also include input data values for those activities that have it. |
| Statistics           | <p>Every <i>period</i> minutes, where <i>period</i> is the length of time over which statistics are gathered. This period is defined in the WLM_COLLECT_INT database configuration parameter.</p> <p>Data is also collected when the WLM_COLLECT_STATS stored procedure is called.</p>                                                                                                                  | Statistics computed from the activities that executed within each service class, workload, or work class that exists on the system.                                                                                                                                                                       |
| Threshold violations | Upon detection of a threshold violation.                                                                                                                                                                                                                                                                                                                                                                | Threshold violation information.                                                                                                                                                                                                                                                                          |

**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 overhead this event monitor incurs by dropping it.

## Collecting information about database system events

Event monitors are used to collect information about the database and any connected applications when specified events occur. Event monitors are database objects, and as such, they are created and manipulated using SQL data definition language (SQL DDL) statements.

You will need DBADM authority to create and manipulate event monitors.

The steps listed below represent a typical life cycle of an event monitor. These steps need not necessarily be executed in the presented order, if at all. For instance, depending on usage it is possible that an event monitor is never dropped or even deactivated. There are, however, two constants in an event monitor's life cycle: the first step will always be the creation of the event monitor and the last step will always be the deletion of the event monitor.

1. Creating an event monitor.
2. For file and pipe event monitors only:
  - :Ensure that the directory or named pipe that will receive the event records exists. The event monitor will not activate otherwise.

On AIX<sup>®</sup>, you can create named pipes by using the mkfifo command. On Linux<sup>®</sup> and other UNIX types (such as the Solaris operating system) use the pipe() routine.

On Windows<sup>®</sup>, you can create named pipes by using the CreateNamedPipe() routine.

- *For pipe event monitors only:* Open the named pipe prior to activating the event monitor. This can be done with an operating system function:
  - for UNIX: open()
  - for Windows: ConnectNamedPipe()

This can also be done with the db2evmon executable:

```
db2evmon -db databasename
 -evm eventmonname
```

*databasename* represents the name of the database being monitored.

*evmonname* represents the name of the event monitor.

3. Activate the newly created event monitor to enable it to collect information.

```
SET EVENT MONITOR evmonname STATE 1;
```

If the event monitor was created with the AUTOSTART option, the event monitor will be activated when the first user connects to the database. Furthermore, once an event monitor has been explicitly activated, it will automatically restart whenever the database is reactivated. Restarting occurs until the event monitor is explicitly deactivated or until the instance is stopped. When a table event monitor is started, the event monitor updates the *evmon\_activates* column of the SYSCAT.EVENTMONITORS catalog table. This change is logged, so the DATABASE CONFIGURATION will display:

```
Database is consistent = 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 will be produced.

4. To see if an event monitor is active or inactive, issue the SQL function EVENT\_MON\_STATE in a query against the table, SYSCAT.EVENTMONITORS:

```
SELECT evmonname, EVENT_MON_STATE(evmonname) FROM
 syscat.eventmonitors;
```

A list of all existing event monitors will be listed, along with their status. A returned value of 0 indicates that the specified event monitor is inactive, and 1 indicates that it is active.

5. Read event monitor output. For write-to-table event monitors this involves examining the target tables. To access file or pipe event monitor data from the CLP see the Related task, Formatting file or pipe event monitor output from a command line".
6. To deactivate, or turn off an event monitor, use the SET EVENT MONITOR statement:

```
SET EVENT MONITOR evmonname STATE 0
```

Deactivating an event monitor does not result in its deletion. It will exist as a dormant database object. Deactivating an event monitor will flush all its contents. Hence, if you reactivate a deactivated event monitor, it will only contain information collected since its reactivation.

7. After you deactivate a pipe event monitor, close the corresponding named pipe. In UNIX use the close() function, and in Windows 2000 use the DisconnectNamedPipe() function.

8. To eliminate an event monitor object, use the DROP EVENT MONITOR statement:

```
DROP EVENT MONITOR evmonname
```

You can only drop an event monitor if it is inactive.

9. After you drop a pipe event monitor, delete the corresponding named pipe. In UNIX use the `unlink()` function, and in Windows use the `CloseHandle()` function. When dropping a write-to-table event monitor, the associated target tables are not dropped. Similarly, when dropping a file event monitor, the associated files are not deleted.

---

## Creating an event monitor

The first step in an event monitor's life cycle is its creation. Before you create an event monitor, you must determine where the event records are to be sent: to SQL tables, files, or through named pipes.

You will need DBADM authority to create an event monitor.

For each event record destination there are particular options that are to be specified in the CREATE EVENT MONITOR SQL statement. The target table of a CREATE EVENT MONITOR statement must be a non-partitioned table. Monitoring events in a partitioned database also requires special attention.

1. Create a table event monitor.
2. Create a file event monitor.
3. Create a pipe event monitor.
4. Create an event monitor for a partitioned database.

Once an event monitor is created and activated, it will record monitoring data as its specified events occur.

## Creating a table event monitor

When creating an event monitor you must determine where the information it collects is to be stored. A table event monitor streams event records to SQL tables, presenting a simple alternative to file and pipe event monitors in enabling you to easily capture, parse, and manage event monitoring data. For every event type an event monitor collects, target tables are created for each of the associated logical data groups.

You will need DBADM authority to create a table event monitor.

The target table of a CREATE EVENT MONITOR statement must be a non-partitioned table.

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 `db2evtbl` command. Simply provide the name of the event monitor and the desired 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 CLP.

1. Indicate that event monitor data is to be collected in a table (or set of tables).

```
CREATE EVENT MONITOR d1mon FOR eventtype
WRITE TO TABLE
```

dlmon is the name of the event monitor.

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 TABLE
```

This event monitor will monitor for the CONNECTIONS and DEADLOCKS WITH DETAILS event types. Assuming that the above statement was issued by the user 'riihi', the derived names and table spaces of the target tables are as follows:

- riihi.connheader\_dlmon
- riihi.conn\_dlmon
- riihi.connmemuse\_dlmon
- riihi.deadlock\_dlmon
- riihi.dlconn\_dlmon
- riihi.dllock\_dlmon
- riihi.control\_dlmon

3. Specify the size of the table event monitor buffers (in 4K pages) by adjusting the BUFFERSIZE value:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
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.

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 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 dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE 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 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 performance overhead from event monitoring:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
```

5. Indicate the logical data groups from which you need to collect event records. Event monitors store the data from each logical data group in corresponding tables.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE CONN, DLCONN, DLLOCK
BUFFERSIZE 8 NONBLOCKED
```

The CONN, DLCONN, and DLLOCK logical data groups are selected. Not mentioned are the other available logical data groups, CONNHEADER, DEADLOCK, or CONTROL. Data relevant to CONNHEADER, DEADLOCK, or CONTROL will not be stored for the dlmon event monitor.

6. Indicate the monitor elements for which you need to collect data.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE CONN,
DLCONN (EXCLUDES(agent_id, lock_wait_start_time)),
DLLOCK (INCLUDES(lock_mode, table_name))
BUFFER SIZE 8 NONBLOCKED
```

All the monitor elements for CONN are captured (this is the default behavior). For DLCONN, all monitor elements except **agent\_id** and **lock\_wait\_start\_time** are captured. And finally, for DLLOCK, **lock\_mode** and **table\_name** are the only monitor elements captured.

7. Provide names for the tables to be created, and designate a table space:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE CONN,
DLCONN (TABLE mydept.dlconnections
EXCLUDES(agent_id, lock_wait_start_time)),
DLLOCK (TABLE dllocks IN mytablespace
INCLUDES(lock_mode, table_name))
BUFFER SIZE 8 NONBLOCKED
```

Assuming that the above statement was issued by the user riihi, the derived names and table spaces of the target tables are as follows:

- CONN: riihi.conn\_dlmon (on the default table space)
- DLCONN: mydept.dlconnections (on the default table space)
- DLLOCK: riihi.dllocks (on the MYTABLESPACE table space)

The default table space is assigned from IBMDEFAULTGROUP, provided the event monitor definer has USE privileges. If the definer does not have USE privileges over this table space, then a table space over which the definer does have USE privileges will be assigned.

8. Indicate how full the table space can get before the event monitor automatically deactivates:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE DLCONN PCTDEACTIVATE 90
BUFFER SIZE 8 NONBLOCKED
```

When the table space reaches 90% capacity, the dlmon 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.

9. 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 TABLE BUFFER SIZE 8 NONBLOCKED
AUTOSTART NONBLOCKED
```

- To create an event monitor that does not start automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
MANUALSTART
```

10. To activate or deactivate an event monitor, use the SET EVENT MONITOR STATE statement.

Once a table event monitor is created and activated, it will record monitoring data as its specified events occur.

## Event monitor table management

You can define an event monitor to store its event records in SQL tables. To do this, use the CREATE EVENT MONITOR statement with the WRITE TO TABLE clause.

Upon the creation of a write-to-table event monitor, the database creates *target tables* to store records for each of the logical data groups returning data. By default, the database creates the tables in the event monitor creator's schema, and names the tables according to their corresponding logical data group and event monitor name. In each table, the column names match the monitor element names that they represent.

For example, the user riihi is creating an event monitor that captures STATEMENTS events:

```
CREATE EVENT MONITOR foo 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. The database created the following tables:

- riihi.connheader\_foo
- riihi.stmt\_foo
- riihi.subsection\_foo
- riihi.control\_foo

In addition to the tables representing logical data groups specific to individual event types, a control table is created for every write-to-table event monitor. This is represented above by riihi.control\_foo. A control table contains event monitor metadata, specifically, from the event\_start, event\_db\_header (**conn\_time** monitor element only), and event\_overflow logical data groups.

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.

Write-to-table event monitor target tables must be pruned manually. On highly active systems, event monitors can quickly fill machine space due to the high volume of data they record. Unlike event monitors that write to files or named pipes, you can define write-to-table event monitors to record only certain logical data groups, or monitor elements. This feature enables you to collect only the data relevant to your purposes and reduce the volume of data generated by the event monitors. For example, the following statement defines an event monitor that captures TRANSACTIONS events, but only from the event\_xact logical data group, and including only the **lock\_escal** monitor element:

```
CREATE EVENT MONITOR foo_lite FOR TRANSACTIONS WRITE TO TABLE
XACT(INCLUDES(lock_escal))
```

There are circumstances where it may not be desirable to have the event monitor's target tables residing in the default schema, with default table names, in the default table space. For instance, you may want the target tables to exist in their own table space if you are anticipating high volumes of monitoring data.

You can specify the schema, table, and table space names in the CREATE EVENT MONITOR statement. The schema name is provided along with the table name, forming a derived name for the table.

A target table can only be used by a single event monitor. If a target table is found to already be defined for another event monitor, or if it cannot be created for any other reason, the CREATE EVENT MONITOR statement will fail.

The table space name can be added after the table name with the optional IN clause. Unlike the target tables, which DB2 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 will only be active on database partitions where the table space containing the event monitor table exists. When 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.log file.

For increased performance in retrieving event monitor data, you can create indexes for the event tables. You can also add additional table attributes, such as triggers, relational integrity, and constraints. The event monitor will ignore 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 foo FOR STATEMENTS
WRITE TO TABLE CONNHEADER,
STMT (TABLE mydept.statements),
SUBSECTION (TABLE subsections, IN mytablespace)
```

Assuming that the above statement was issued by the user 'riihi', the derived names and table spaces of the target tables are as follows:

- CONNHEADER: riihi.connheader\_foo (on the default table space)
- STMT: mydept.statements (on the default table space)
- SUBSECTION: riihi.subsections (on 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 at which percentage of table space capacity the event monitor will deactivate. This can be declared in the PCTDEACTIVATE clause in the CREATE EVENT MONITOR statement.



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.

The following table presents the default target table names as sorted by the event type for which they are returned.

*Table 10. Write-to-Table Event Monitor Target Tables*

| Event type                            | Target table names | Available information                               |
|---------------------------------------|--------------------|-----------------------------------------------------|
| DEADLOCKS                             | CONNHEADER         | Connection metadata                                 |
|                                       | DEADLOCK           | Deadlock data                                       |
|                                       | DLCONN             | Applications and locks involved in deadlock         |
|                                       | CONTROL            | Event monitor metadata                              |
| DEADLOCKS WITH DETAILS                | CONNHEADER         | Connection metadata                                 |
|                                       | DEADLOCK           | Deadlock data                                       |
|                                       | DLCONN             | Applications involved in deadlock                   |
|                                       | DLLOCK             | Locks involved in deadlock                          |
|                                       | CONTROL            | Event monitor metadata                              |
| DEADLOCKS WITH DETAILS HISTORY        | CONNHEADER         | Connection metadata                                 |
|                                       | DEADLOCK           | Deadlock data                                       |
|                                       | DLCONN             | Applications involved in deadlock                   |
|                                       | DLLOCK             | Locks involved in deadlock                          |
|                                       | STMTHIST           | List of the previous statements in the unit of work |
| DEADLOCKS WITH DETAILS HISTORY VALUES | CONNHEADER         | Connection metadata                                 |
|                                       | DEADLOCK           | Deadlock data                                       |
|                                       | DLCONN             | Applications involved in deadlock                   |
|                                       | DLLOCK             | Locks involved in deadlock                          |
|                                       | STMTHIST           | List of the previous statements in the unit of work |
|                                       | STMTVALS           | Input Data values of statements in STMTHIST table   |
|                                       | CONTROL            | Event monitor metadata                              |
| STATEMENT                             | CONNHEADER         | Connection metadata                                 |
|                                       | STMT               | Statement data                                      |
|                                       | SUBSECTION         | Statement data specific to subsection               |
|                                       | CONTROL            | Event monitor metadata                              |
| TRANSACTION                           | CONNHEADER         | Connection metadata                                 |
|                                       | XACT               | Transaction data                                    |
|                                       | CONTROL            | Event monitor metadata                              |
| CONNECTIONS                           | CONNHEADER         | Connection metadata                                 |
|                                       | CONN               | Connection data                                     |
|                                       | CONTROL            | Event monitor metadata                              |
|                                       | CONMEMUSE          | Memory pool metadata                                |



Table 10. Write-to-Table Event Monitor Target Tables (continued)

| Event type           | Target table names  | Available information                                                                                                                                                                         |
|----------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DATABASE             | DB                  | Database manager data                                                                                                                                                                         |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
|                      | DBMEMUSE            | Memory pool metadata                                                                                                                                                                          |
| BUFFERPOOLS          | BUFFERPOOL          | Buffer pool data                                                                                                                                                                              |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
| TABLESPACES          | TABLESPACE          | Tablespace data                                                                                                                                                                               |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
| TABLES               | TABLE               | Table data                                                                                                                                                                                    |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
| ACTIVITIES           | ACTIVITY            | Activities that completed executing or were captured in progress.                                                                                                                             |
|                      | ACTIVITYSTMT        | Statement information for activities that are statements.                                                                                                                                     |
|                      | ACTIVITYVALS        | Input data values for activities that have them. The data types that can be reported excludes the following: CLOB, REF, BOOLEAN, STRUCT, DATALINK, LONG VARGRAPHIC, LONG, XMLLOB, and DBCLOB. |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
| STATISTICS           | SCSTATS             | Statistics computed from the activities that executed within each service class, work class, or workload in the system.                                                                       |
|                      | WCSTATS             |                                                                                                                                                                                               |
|                      | WLSTATS             |                                                                                                                                                                                               |
|                      | HISTOGRAMBIN        |                                                                                                                                                                                               |
|                      | QSTATS              |                                                                                                                                                                                               |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |
| THRESHOLD VIOLATIONS | THRESHOLDVIOLATIONS | List of thresholds that have been violated as well as the times of violations.                                                                                                                |
|                      | CONTROL             | Event monitor metadata                                                                                                                                                                        |

The following logical data groups are not collected for write-to-table event monitors:

- event\_log\_stream\_header
- event\_log\_header
- event\_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 is a set of data type mappings that correspond the original system monitor data types of the monitor elements (found in sqlmon.h) to the SQL data types of the table columns.

Table 11. 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            |

Table 11. System Monitor Data Type Mappings (continued)

| System monitor data type             | SQL data type     |
|--------------------------------------|-------------------|
| 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 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.

You will need DBADM authority to create a file event monitor.

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.

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 dlmon FOR eventtype
 WRITE TO FILE '/tmp/dlevents'
```

dlmon is the name of the event monitor.

/tmp/dlevents is the name of the directory path (on UNIX) 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 performance overhead from 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' BUFFERSIZE 8
NONBLOCKED MANUALSTART
```
8. To activate or deactivate an event monitor, use the SET EVENT MONITOR STATE statement.

Once a file event monitor is created and activated, it will record monitoring data as its specified events occur.

## Event monitor file management

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. This directory will not be created by DB2 if it does not exist. Before the monitor is activated, the directory must exist, or the SET EVENT MONITOR command will return an error. When a file event monitor is first activated, a control file named db2event.ctl 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 automatically directed to the next file. 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.
```

You can avoid this situation by removing full files. Any event file except the active file can be removed while the event monitor is still running.

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.

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`.

You may 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. In order to keep track of which events have been processed in the active file, you can create an application that simply keeps track of the file number and location of the last record processed. When processing the trace the next time around, the application can simply seek to that file location.

## Write-to-table and file event monitor buffering

The event monitor process buffers its records, using two internal buffers, before writing them to a file or table. Records are written 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.

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 overhead, 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 could occur if you are capturing the `stmt_text` monitor element and applications attached to the database being monitored issue lengthy SQL statements. If you need to capture all of the event record information, specify larger buffers. Keep in mind that larger buffers will result in less frequent writes to file or table.

## 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.

You will need DBADM authority to create a pipe event monitor.

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 instance, if it is full), monitor data will be lost.

Pipe event monitors are defined with the CREATE EVENT MONITOR statement.

1. Indicate that event monitor data is to be directed to a named pipe.

```
CREATE EVENT MONITOR dlmon FOR eventtype
 WRITE TO PIPE '/home/riihi/dlevents'
```

dlmon is the name of the event monitor.

/home/riihi/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 prior to 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 dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
 WRITE TO PIPE '/home/riihi/dlevents'
```

This event monitor will monitor for the CONNECTIONS and DEADLOCKS WITH DETAILS 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 dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
 WRITE TO PIPE '/home/riihi/dlevents'
 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 PIPE '/home/riihi/dlevents'
 MANUALSTART
```

4. To activate or deactivate an event monitor, use the SET EVENT MONITOR STATE statement.

Once a pipe event monitor is created and activated, it will record monitoring data as its specified events occur.

## Event monitor named pipe management

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, 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.

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.

## Creating an event monitor for partitioned databases

When creating a file or pipe event monitor on partitioned database systems you need to determine the scope of the monitoring data you wish to collect.

You will need DBADM authority to create event monitors for partitioned databases.

An event monitor uses an operating system process or a thread to write the event records. The database partition where this process or thread runs is called the monitor partition. File and pipe event monitors can be monitoring events as they occur locally on the monitor partition, or globally as they occur on any partition where the DB2 database manager is running. A global event monitor writes a single trace on the monitoring partition that contains activity from all partitions. Whether an event monitor is local or global is referred to as its monitoring scope.

Both the monitor partition and monitor scope are specified with the `CREATE EVENT MONITOR` statement.

An event monitor can only be activated if the monitor partition is active. If the `SET EVENT MONITOR` statement is used to activate an event monitor but the monitor partition is not yet active, then event monitor activation will occur when the monitor partition is next started. Furthermore, event monitor activation will automatically occur until the event monitor is explicitly deactivated or the instance is explicitly deactivated. For example, on database partition 0:



```
db2 connect to sample
db2 create event monitor foo ... on dbpartitionnum 2
db2 set event monitor foo state 1
```

After running the above commands, event monitor `foo` will activate automatically whenever the database `sample` activates on database partition 2. This automatic activation occurs until `db2 set event monitor foo state 0` is issued, or partition 2 is stopped.

For write-to-table event monitors, the notion of local or global scope is not applicable. When a write-to-table event monitor is activated, an event monitor process runs on all of the partitions. (More specifically, the event monitor process will run on partitions that belong to the database partition groups in which the target tables reside.) Each partition where the event monitor process is running also has the same set of target tables. The data in these tables will be different as it represents the individual partition's view of the monitoring data. You can get aggregate values from all the partitions by issuing SQL statements that access the desired values in each partition's event monitor target tables.

The first column of each target 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 will contain the same value. This means that if a data partition is dropped and data redistribution is performed, all data on the dropped database partition will go to one other database partition instead of being evenly distributed. Therefore, before removing a database partition, consider deleting all table rows on that database partition.

In addition, a column named `PARTITION_NUMBER` can be defined for each table. This column contains the number of the partition on which the data was inserted. Unlike the `PARTITION_KEY` column, the `PARTITION_NUMBER` column is not mandatory.

The table space within which target tables are defined must exist across all partitions that will have event monitor data written to them. Failure to observe this rule will result in records not being written to the log on partitions (with event monitors) where the table space does not exist. Events will still be written on partitions where the table space does exist, and no error will be returned. This behavior allows users to 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 only inserted on the catalog database partition. This requires that the table space for the control table exist on the catalog database partition. If it does not exist on the catalog database partition, these inserts are not performed.

If a partition is not yet active when a write-to-table event monitor is activated, the event monitor will be activated when that partition next activates.

**Note:** The lock list in the detailed deadlock connection event will only contain the locks held by the application on the partition where it is waiting for the lock. For example, if an application involved in a deadlock is waiting for a lock on node 20, only the locks held by that application on node 20 will be included in the list.



1. Specify the partition to be monitored.

```
CREATE EVENT MONITOR dlmon FOR DEADLOCKS
 WRITE TO FILE '/tmp/dlevents'
 ON PARTITION 3
```

dlmon represents the name of the event monitor.

/tmp/dlevents is the name of the directory path (on UNIX) where the event monitor is to write the event files.

3 represents the partition number to be monitored.

2. Specify if the event monitor data is to be collected at a local or global scope. To collect event monitor reports from all partitions issue the following statement:

```
CREATE EVENT MONITOR dlmon FOR DEADLOCKS
 WRITE TO FILE '/tmp/dlevents'
 ON PARTITION 3 GLOBAL
```

Only deadlock and deadlock with details event monitors can be defined as GLOBAL. All partitions will report deadlock-related event records to partition 3.

3. To collect event monitor reports from only the local partition issue the following statement:

```
CREATE EVENT MONITOR dlmon FOR DEADLOCKS
 WRITE TO FILE '/tmp/dlevents'
 ON PARTITION 3 LOCAL
```

This is the default behavior for file and pipe event monitors in partitioned databases. The LOCAL and GLOBAL clauses are ignored for write-to-table event monitors.

4. It is possible to review the monitor partition and scope values for existing event monitors. To do this query the SYSCAT.EVENTMONITORS table with the following statement:

```
SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
```

Once an event monitor is created and activated, it will record monitoring data as its specified events occur.

---

## Event monitor sample output

To illustrate the nature of event monitoring, here is an example of a deadlock monitoring scenario. To implement this scenario you will need three DB2 CLP windows. We will refer to the first CLP as the *Monitor Session*, and the remaining CLPs as *Application 1* and *Application 2*. You will also need the DB2 SAMPLE database.

**Note:** You can create and populate the SAMPLE database with either of the following steps:

- UNIX: sqllib/bin/db2saml
- Windows: sqllib\bin\db2saml.exe

From the Monitor Session, define an event monitor that logs table data and the occurrence of deadlocks between connections to a database:

```
db2 connect to sample
db2 "create event monitor dlmon for tables, deadlocks with details
 write to file 'c:\dlmon'"
mkdir c:\dlmon
db2 "set event monitor dlmon state 1"
```

Now, two applications using the database enter a deadlock. A deadlock is a situation where each application is holding a lock that the other one needs in order to continue processing. The deadlock is eventually detected and resolved by the DB2 deadlock detector component, which will rollback one of the transactions. As a result only one of the applications will successfully complete their transaction. The following figures illustrate this scenario.

### Application 1

```
db2 connect to sample
db2 +c "lock table staff in exclusive mode"
```

**Note:** The +c option used above turns autocommit off for the given CLP command.

Application 1 is now holding an exclusive lock on the STAFF table.

### Application 2

```
db2 connect to sample
db2 +c "lock table department in exclusive mode"
```

Application 2 is now holding an exclusive lock on the DEPARTMENT table.

### Application 1

```
db2 +c "select deptname from department"
```

Assuming cursor stability, Application 1 needs an intent share lock on the department table in order to fetch rows, but it cannot be obtained because Application 2 has an exclusive lock on it. Application 1 enters a LOCK WAIT state, while it waits for the lock to be released.

### Application 2

```
db2 +c "select name from staff"
```

Application 2 also enters a LOCK WAIT state, while waiting for Application 1 to release its exclusive lock on the staff table.

These applications are now in a deadlock. This waiting will never be resolved because each application is holding a resource that the other one needs in order to continue. Eventually, the deadlock detector checks for deadlocks and chooses a victim to rollback:

### Application 2

```
SQLN0991N The current transaction has been
rolled back because of a deadlock or timeout.
Reason code "2". SQLSTATE=40001
```

At this point the event monitor logs a deadlock event to its target. Application 1 can now continue:

### Application 1

```
DEPTNAME

...
PLANNING
INFORMATION CENTER
DEVELOPMENT CENTER
```

...  
BRANCH OFFICE J2

14 record(s) selected.

Because an event monitor buffers its output and this scenario did not generate enough event records to fill a buffer, the event monitor values are forced to the event monitor output writer. In order to generate the table event data (which is generated upon the deactivation of a database) Application 1, Application 2, and the Monitor Session will disconnect from the database.

## Application 1

db2 connect reset

## Application 2

db2 connect reset

After disconnecting from the database in the Monitor Session CLP, the event trace is written as a binary file. It can now be formatted using the db2evmon tool:

## Monitor Session

```
db2 connect reset
db2evmon -path c:\dlmon
```

The logical data groupings used by the event monitor are ordered and presented according to four different levels: Monitor, Prolog, Contents, and Epilog.

## Monitor

Information at the Monitor level is generated for all event monitors. It consists of event monitor meta-data.

## Prolog

The Prolog information is generated when the event monitor is activated.

```

 EVENT LOG HEADER
Event Monitor name: DLMON
Server Product ID: SQL09013
Version of event monitor data: 8
Byte order: LITTLE ENDIAN
Number of nodes in db2 instance: 1
Codepage of database: 1208
Territory code of database: 1
Server instance name: DB2

Database Name: SAMPLE
Database Path: C:\DB2\NODE0000\SQL00001\
First connection timestamp: 04/12/2007 18:07:29.266219
Event Monitor Start time: 04/12/2007 18:08:56.150236

```

## Contents

Information specific to the event monitor's specified event types is presented in the Contents section. Events recorded in this section contain references to the application that spawned them (an application handle or an application ID). If you

are tracking events from multiple applications, use the application identifiers to keep track of the various events. The overflow event, unlike all the others in the Contents section, does not correspond to a specific event type. It tracks the number of records lost: those generated when the system cannot keep up with a (non-blocked) event monitor. In this example, there are deadlock events, spawned by the deadlock state incurred by the previous statements.

3) Deadlock Event ...

Deadlock ID: 1  
Number of applications deadlocked: 2  
Deadlock detection time: 04/12/2007 18:12:14.335861  
Rolled back Appl participant no: 2  
Rolled back Appl Id: \*LOCAL.DB2.070412221044  
Rolled back Appl seq number: : 0001

4) Connection Header Event ...

Appl Handle: 67  
Appl Id: \*LOCAL.DB2.070412221044  
Appl Seq number: 00001  
DRDA AS Correlation Token: \*LOCAL.DB2.070412221044  
Program Name : db2bp.exe  
Authorization Id: ADMINISTRATOR  
Execution Id : ADMINISTRATOR  
Codepage Id: 1252  
Territory code: 1  
Client Process Id: 2412  
Client Database Alias: SAMPLE  
Client Product Id: SQL09013  
Client Platform: Unknown  
Client Communication Protocol: Local  
Client Network Name: CONNOR  
Connect timestamp: 04/12/2007 18:10:44.902756

5) Deadlocked Connection ...

Deadlock ID: 1  
Participant no.: 2  
Participant no. holding the lock: 1  
Appl Id: \*LOCAL.DB2.070412221044  
Appl Seq number: 00001  
Appl Id of connection holding the lock: \*LOCAL.DB2.070412220933  
Seq. no. of connection holding the lock: 00001  
Lock wait start time: 04/12/2007 18:12:09.043884  
Lock Name : 0x02000F0000000000000000000000054  
Lock Attributes : 0x00000000  
Release Flags : 0x00000001  
Lock Count : 1  
Hold Count : 0  
Current Mode : none  
Deadlock detection time: 04/12/2007 18:12:14.336187  
Table of lock waited on : STAFF  
Schema of lock waited on : ADMINISTRATOR  
Data partition id for table : 0  
Tablespace of lock waited on : USERSPACE1  
Type of lock: Table  
Mode of lock: X - Exclusive  
Mode application requested on lock: IS - Intent Share  
Node lock occurred on: 0  
Lock object name: 15  
Application Handle: 67  
Deadlocked Statement:  
Type : Dynamic  
Operation: Fetch  
Section : 201  
Creator : NULLID  
Package : SQLC2F0A  
Cursor : SQLCUR201  
Cursor was blocking: FALSE

```

Text : select name from staff
List of Locks:
Lock Name : 0x010000000100000001002C0056
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Variation
Data partition id : -1
Mode : S - Share

Lock Name : 0x0000050007BE130080955B0343
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Catalog Cache
Data partition id : -1
Mode : S - Share

Lock Name : 0x53514C4332463041F12CF8E241
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Plan
Data partition id : -1
Mode : S - Share

Lock Name : 0x53514C4445464C5428DD630641
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Plan
Data partition id : -1
Mode : S - Share

Lock Name : 0x0200050000000000000000000054
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 255
Hold Count : 0
Lock Object Name : 5
Object Type : Table
Tablespace Name : USERSPACE1
Table Schema : ADMINISTRATOR
Table Name : DEPARTMENT
Data partition id : 0
Mode : X - Exclusive

```

```

Locks Held: 5
Locks in List: 5

```

```

6) Connection Header Event ...
Appl Handle: 66
Appl Id: *LOCAL.DB2.070412220933
Appl Seq number: 00001
DRDA AS Correlation Token: *LOCAL.DB2.070412220933
Program Name : db2bp.exe
Authorization Id: ADMINISTRATOR
Execution Id : ADMINISTRATOR
Codepage Id: 1252

```

Territory code: 1  
Client Process Id: 2256  
Client Database Alias: SAMPLE  
Client Product Id: SQL09013  
Client Platform: Unknown  
Client Communication Protocol: Local  
Client Network Name: CONNOR  
Connect timestamp: 04/12/2007 18:09:33.854626

7) Deadlocked Connection ...

Deadlock ID: 1  
Participant no.: 1  
Participant no. holding the lock: 2  
Appl Id: \*LOCAL.DB2.070412220933  
Appl Seq number: 00001  
Appl Id of connection holding the lock: \*LOCAL.DB2.070412221044  
Seq. no. of connection holding the lock: 00001  
Lock wait start time: 04/12/2007 18:11:52.490288  
Lock Name : 0x02000500000000000000000000054  
Lock Attributes : 0x00000000  
Release Flags : 0x00000001  
Lock Count : 1  
Hold Count : 0  
Current Mode : none  
Deadlock detection time: 04/12/2007 18:12:14.386492  
Table of lock waited on : DEPARTMENT  
Schema of lock waited on : ADMINISTRATOR  
Data partition id for table : 0  
Tablespace of lock waited on : USERSPACE1  
Type of lock: Table  
Mode of lock: X - Exclusive  
Mode application requested on lock: IS - Intent Share  
Node lock occurred on: 0  
Lock object name: 5  
Application Handle: 66  
Deadlocked Statement:  
Type : Dynamic  
Operation: Fetch  
Section : 201  
Creator : NULLID  
Package : SQLC2F0A  
Cursor : SQLCUR201  
Cursor was blocking: FALSE  
Text : select deptname from department

List of Locks:

|                   |                                |
|-------------------|--------------------------------|
| Lock Name         | : 0x01000000010000000100620056 |
| Lock Attributes   | : 0x00000000                   |
| Release Flags     | : 0x40000000                   |
| Lock Count        | : 1                            |
| Hold Count        | : 0                            |
| Lock Object Name  | : 0                            |
| Object Type       | : Internal - Variation         |
| Data partition id | : -1                           |
| Mode              | : S - Share                    |
| Lock Name         | : 0x0000050006FC1F0000885B0343 |
| Lock Attributes   | : 0x00000000                   |
| Release Flags     | : 0x40000000                   |
| Lock Count        | : 1                            |
| Hold Count        | : 0                            |
| Lock Object Name  | : 0                            |
| Object Type       | : Internal - Catalog Cache     |
| Data partition id | : -1                           |
| Mode              | : S - Share                    |
| Lock Name         | : 0x53514C4332463041F12CF8E241 |
| Lock Attributes   | : 0x00000000                   |

```

Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Plan
Data partition id : -1
Mode : S - Share

Lock Name : 0x53514C4445464C5428DD630641
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 1
Hold Count : 0
Lock Object Name : 0
Object Type : Internal - Plan
Data partition id : -1
Mode : S - Share

Lock Name : 0x02000F000000000000000000000054
Lock Attributes : 0x00000000
Release Flags : 0x40000000
Lock Count : 255
Hold Count : 0
Lock Object Name : 15
Object Type : Table
Tablespace Name : USERSPACE1
Table Schema : ADMINISTRATOR
Table Name : STAFF
Data partition id : 0
Mode : X - Exclusive

```

```

Locks Held: 5
Locks in List: 5

```

## Epilog

The Epilog information is generated during database deactivation (when the last application finishes disconnecting):

### 8) Table Event ...

```

Table schema: SYSIBM
Table name: SYSTABLES
Data partition id: 0

```

```

Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 45
Index object pages: 20
Lob object pages: 448
Long object pages: 0
Rows read: 2
Rows written: 0
Overflow Accesses: 1
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 04/12/2007 18:14:36.364389

```

### 9) Table Event ...

```

Table schema: ADMINISTRATOR
Table name: DEPARTMENT
Data partition id: 0

```

```

Record is the result of a flush: FALSE
Table type: User
Data object pages: 1
Index object pages: 5
Lob object pages: 0

```

```
Long object pages: 0
Rows read: 14
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 2
Table event timestamp: 04/12/2007 18:14:36.364389
```

## 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).

No authorization is required unless you are connecting to the database, in which case one of the following is required:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM

You can indicate which event monitor's output you will format by either providing the path of the event files, or providing the name of the database and the event monitor's name. To format event monitor output:

- Specify the directory containing the event monitor files:

```
db2evmon -path '/tmp/dlevents'
```

`/tmp/dlevents` 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.

## Event records and their corresponding applications

In an event trace for an active database with hundreds of attached applications, it can be tedious to track event records associated with a specific application. For traceability, each event record includes the application handle and application ID. These allow you to correlate each record with the application for which the event record was generated.

The application handle (**agent\_id**) is unique system-wide for the duration of the application. However, it will eventually be reused (a 16 bit counter is used to generate this identifier -- on partitioned database systems this consists of the coordinating partition number and a 16 bit counter). In most cases, this reuse is not a problem, since an application reading records from the trace is able to detect a connection that was terminated. For example, encountering (in the trace) a connection header with a known `agent_ID` implies that the previous connection with this `agent_ID` was terminated.

The application ID is a string identifier that includes a timestamp and is guaranteed to remain unique, even after stopping and restarting the database manager.



Finding event records for a certain application is particularly easy with write-to-table event monitors. In the event monitor tables, where each row corresponds to an event record, the application handle and application ID are default column values. To find all the event records for a given application, you can simply issue an SQL select statement for all event records corresponding to the particular application ID.

## Event monitor self-describing data stream

The output of an event monitor is 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 3 shows the structure of the data stream and Table 12 on page 72 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.

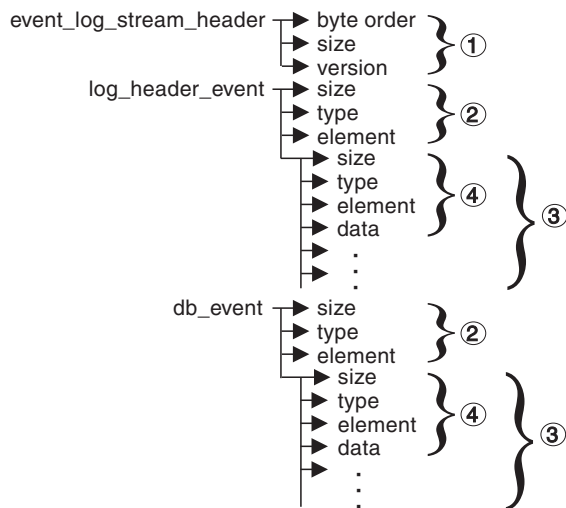


Figure 3. Event Monitor Data Stream

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                                                                                                          |
|-------------------------|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| event_log_stream_header | sqlm_little_endian                         | Not used (for compatibility with previous releases).                                                                 |
|                         | 200                                        | Not used (for compatibility with previous releases).                                                                 |
|                         | sqlm_dbmon_version9_5                      | The version of the database manager that returned the data. Event monitors write data in the self-describing format. |
| log_header_event        | 100                                        | Size of the logical data group.                                                                                      |
|                         | header                                     | Indicates the start of a logical data group.                                                                         |
|                         | log_header                                 | Name of the logical data group.                                                                                      |
|                         | 4                                          | Size of the data stored in this monitor element.                                                                     |
|                         | u32bit                                     | Monitor element type - 32 bit numeric.                                                                               |
|                         | byte_order                                 | The name of the monitor element collected.                                                                           |
|                         | little_endian                              | The collected value for this element.                                                                                |
|                         | 2                                          | Size of the data stored in this monitor element.                                                                     |
|                         | u16bit                                     | Monitor element type - unsigned 16 bit numeric.                                                                      |
| codepage_id             | The name of the monitor element collected. |                                                                                                                      |
| 850                     | The collected value for this element.      |                                                                                                                      |
| db_event                | 100                                        | Size of the logical data group.                                                                                      |
|                         | header                                     | Indicates the start of a logical data group.                                                                         |
|                         | db_event                                   | Name of the logical data group.                                                                                      |
|                         | 4                                          | Size of the data stored in this monitor element.                                                                     |
|                         | u32bit                                     | Monitor element type - unsigned 32 bit numeric.                                                                      |
|                         | lock_waits                                 | The name of the monitor element collected.                                                                           |
|                         | 2                                          | The collected value for this element.                                                                                |

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 numerical 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.

## Transferring event monitor data between systems

When transferring event monitor information between systems using different conventions for storing numerical values, conversions must be made. Information on UNIX platforms is stored in little endian byte order, and information on Windows platforms is stored in big endian byte order. If event monitor data from a little endian source is to be read on a big endian platform, or vice versa, byte conversion is necessary.

1. To convert the numeric values in logical data group headers and monitor elements use the following logic (presented in C):

```
#include sqlmon.h
#define SWAP2(s) (((s) >> 8) & 0xFF) | (((s) << 8) & 0xFF00)

#define SWAP4(l) (((l) >> 24) & 0xFF) | (((l) & 0xFF0000) >> 8) & 0xFF00 \
 | (((l) & 0xFF00) << 8) | ((l) << 24)

#define SWAP8(where)
{
 sqluint32 temp;
 temp = SWAP4(*(sqluint32 *) (where));
 *(sqluint32 *) (where) = SWAP4(* ((sqluint32 *) (where)) + 1);
 * ((sqluint32 *) (where)) + 1 = temp;
}

int HeaderByteReverse(sqlm_header_info * pHeader)
{
 int rc = 0;

 pHeader->size = SWAP4(pHeader->size);
 pHeader->type = SWAP2(pHeader->type);
 pHeader->element = SWAP2(pHeader->element);

 return rc;
}

int DataByteReverse(char * dataBuf, sqluint32 dataSize)
{
 int rc = 0;

 sqlm_header_info * pElemHeader = NULL;
 char * pElemData = NULL;
 sqluint32 dataOffset = 0;
 sqluint32 elemDataSize = 0;
 sqluint32 elemHeaderSize = sizeof(sqlm_header_info);

 // For each of the elements in the data stream that are numeric,
 // perform byte reversal.

 while(dataOffset < dataSize)
 {
 /* byte reverse the element header */
 pElemHeader = (sqlm_header_info *)
 (dataBuf + dataOffset);

 rc = HeaderByteReverse(pElemHeader);
 if(rc != 0) return rc;
 // Remember the element data's size...it will be byte reversed
 // before we skip to the next element.
 elemDataSize = pElemHeader->size;

 /* byte reverse the element data */
 pElemData = (char *)
 (dataBuf + dataOffset + elemHeaderSize);

 if(pElemHeader->type == SQLM_TYPE_HEADER)
 {
 rc = DataByteReverse(pElemData, pElemHeader->size);
 if(rc != 0) return rc;
 }
 }
}
```

```

else
{
switch(pElemHeader->type)
{
case SQLM_TYPE_16BIT:
case SQLM_TYPE_U16BIT:
*(sqluint16 *) (pElemData) =
SWAP2(*(short *) (pElemData));

break;
case SQLM_TYPE_32BIT:
case SQLM_TYPE_U32BIT:
*(sqluint32 *) (pElemData) =
SWAP4(*(sqluint32 *) (pElemData));

break;
case SQLM_TYPE_64BIT:
case SQLM_TYPE_U64BIT:
SWAP8(pElemData);
break;
default:
// Not a numeric type. Do nothing.
break;
}
}
dataOffset = dataOffset + elemHeaderSize + elemDataSize;
}

return 0;
} /* end of DataByteReverse */

```

2. To convert the numeric values in logical data group headers and monitor elements use the following logic (presented in C):

```

#include sqlmon.h
#define SWAP2(s) (((s) >> 8) & 0xFF) | (((s) << 8) & 0xFF00)

#define SWAP4(l) (((l) >> 24) & 0xFF) | (((l) & 0xFF0000) >> 8) & 0xFF00 \
| (((l) & 0xFF00) << 8) | ((l) << 24)

#define SWAP8(where)
{
sqluint32 temp;
temp = SWAP4(*(sqluint32 *) (where));
*(sqluint32 *) (where) = SWAP4(*(sqluint32 *) (where) + 1);
*(sqluint32 *) (where) + 1 = temp;
}

int HeaderByteReverse(sqlm_header_info * pHeader)
{
int rc = 0;

pHeader->size = SWAP4(pHeader->size);
pHeader->type = SWAP2(pHeader->type);
pHeader->element = SWAP2(pHeader->element);

return rc;
}

int DataByteReverse(char * dataBuf, sqluint32 dataSize)
{
int rc = 0;

sqlm_header_info * pElemHeader = NULL;
char * pElemData = NULL;
sqluint32 dataOffset = 0;
sqluint32 elemDataSize = 0;
sqluint32 elemHeaderSize = sizeof(sqlm_header_info);

// For each of the elements in the datas tream that are numeric,
// perform byte reversal.

while(dataOffset < dataSize)
{ /* byte reverse the element header */

```

```

pElemHeader = (sqlm_header_info *)
 (dataBuf + dataOffset);

rc = HeaderByteReverse(pElemHeader);
if(rc != 0) return rc;
// Remember the element data's size...it will be byte reversed
// before we skip to the next element.
elemDataSize = pElemHeader->size;

/* byte reverse the element data */
pElemData = (char *)
 (dataBuf + dataOffset + elemHeaderSize);

if(pElemHeader->type == SQLM_TYPE_HEADER)
{ rc = DataByteReverse(pElemData, pElemHeader->size);
 if(rc != 0) return rc;
}
else
{ switch(pElemHeader->type)
 { case SQLM_TYPE_16BIT:
 case SQLM_TYPE_U16BIT:
 *(sqluint16 *) (pElemData) =
 SWAP2(*(short *) (pElemData));

 break;
 case SQLM_TYPE_32BIT:
 case SQLM_TYPE_U32BIT:
 *(sqluint32 *) (pElemData) =
 SWAP4(*(sqluint32 *) (pElemData));

 break;
 case SQLM_TYPE_64BIT:
 case SQLM_TYPE_U64BIT:
 SWAP8(pElemData);
 break;
 default:
 // Not a numeric type. Do nothing.
 break;
 }
 dataOffset = dataOffset + elemHeaderSize + elemDataSize;
}

return 0;
} /* end of DataByteReverse */

```



---

## Chapter 5. Activity Monitor overview

Use the Activity Monitor to monitor application performance and concurrency, resource consumption, and SQL statement usage of a database or database partition. The Activity Monitor provides a set of predefined reports based on a specific subset of monitor data. These reports allow you to focus monitoring on application performance, application concurrency, resource consumption, and SQL statement use. The Activity Monitor also provides recommendations for most reports. These recommendations can assist you to diagnose the cause of database performance problems, and to tune queries for optimal utilization of database resources.

Figure 4 on page 78 describes the process for using the Activity monitor to solve a problem.

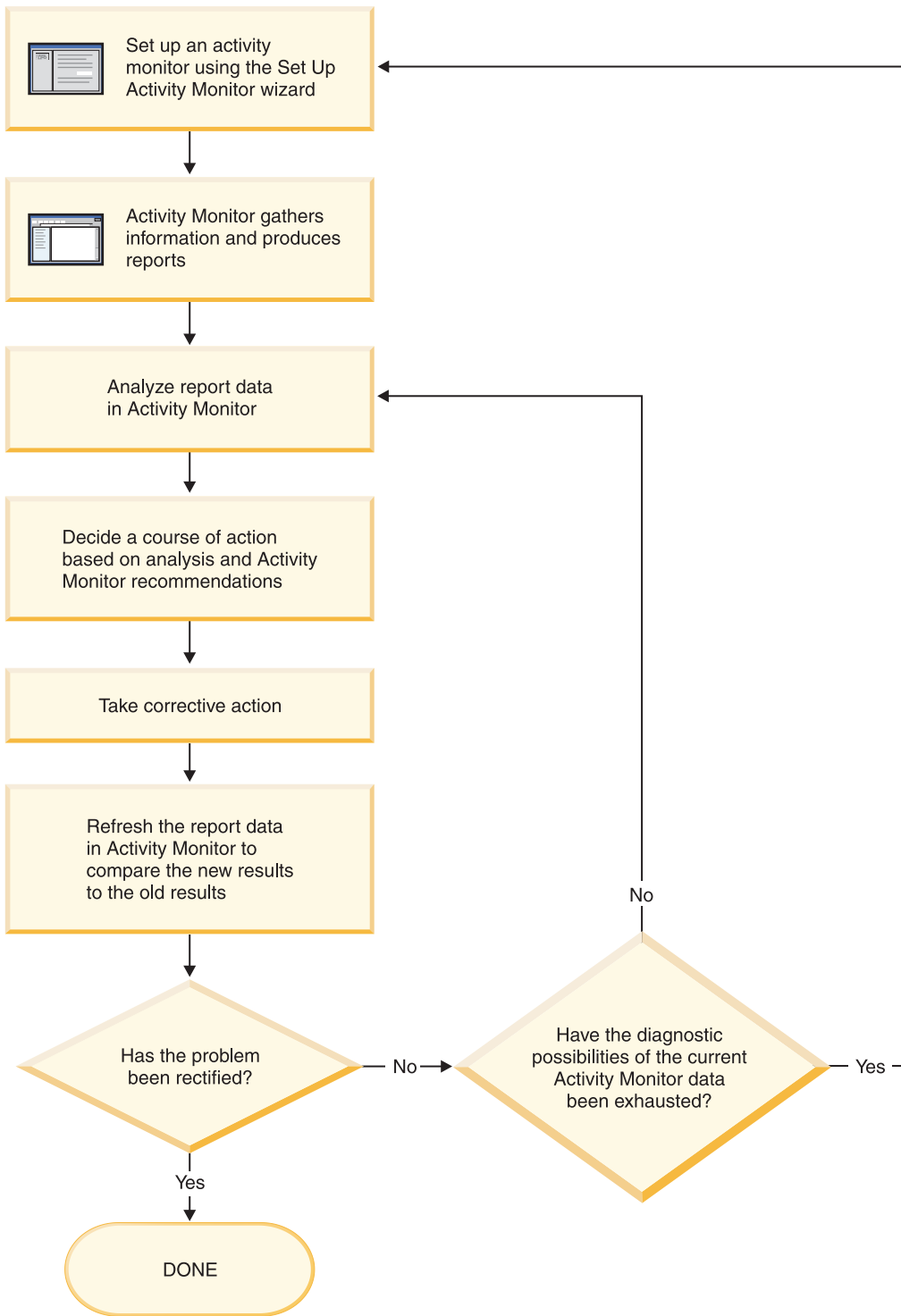


Figure 4. Activity Monitor overview

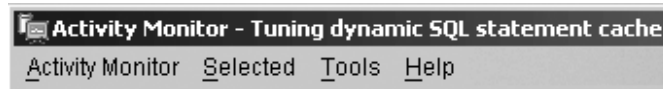


Table 13. Tasks that you can perform from the Activity Monitor

| Tasks from the Activity Monitor      | Aspects of tasks                                                                                          | Invocation                                                                                                                                                                                                                                        |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transactions                         | View transactions running on a selected application.                                                      | Select one or more applications in the <b>Report data</b> pane. Right-click and select <b>Show Latest Transactions</b> . The Application Transactions window opens.                                                                               |
| Statements                           | View SQL statements running on a selected application.                                                    | Select one or more applications in the <b>Report data</b> pane. Right-click and select <b>Show Latest Statements</b> . The Application Statements window opens.                                                                                   |
|                                      | View the text of SQL statements running on a selected application.                                        | From the Application Statements window, right-click on a statement in the <b>Report data</b> pane. Select <b>Show Statement Text</b> .                                                                                                            |
| Application Lock Chains              | View locks and lock-waiting situations that currently affect a selected application.                      | Select an application in the <b>Report data</b> pane. Right-click and select <b>Show Lock Chains</b> . The Application Lock Chains window opens.                                                                                                  |
|                                      | View information about a selected application for which you are viewing lock information.                 | From the Application Lock Chains window, right-click an application, and select <b>About</b> .                                                                                                                                                    |
|                                      | View information about the locks held and the locks waited on by a selected application in your database. | From the Application Lock Chains window, right-click an application, and select <b>Show Lock Details</b> .                                                                                                                                        |
| View report data and recommendations | View information to help you interpret report data.                                                       | From an Activity Monitor window, an Application Statements window, or an Application Transactions window, use the <b>Report</b> arrow to select the report and click the <b>Report Details</b> push button. View the <b>Details</b> page.         |
|                                      | View recommendations provided by Activity Monitor                                                         | From an Activity Monitor window, an Application Statements window, or an Application Transactions window, use the <b>Report</b> arrow to select the report and click the <b>Report Details</b> push button. View the <b>Recommendations</b> page. |

The Activity Monitor interface has several elements that help you organize and interpret the monitor data that is collected:

## Menu bar



Use the menu bar to work with objects in the Activity Monitor, open other administration centers and tools, and access online help.

## Activity Monitor toolbar



Use the toolbar icons to open DB2 tools and view DB2 information.

## Report data pane

| Application Handle (agent ID) | Application Name | Authorization ID | Application ID   | Total CPU Time | User CPU Time |
|-------------------------------|------------------|------------------|------------------|----------------|---------------|
| 18                            | acmerpt.exe      | EDWARDL          | *LOCAL.DB2.00... | 180259         | 10014         |
| 20                            | db2cc.exe        | DB2ADMIN         | *LOCAL.DB2.00... | 30042          | 10014         |
| 22                            | acmefin.exe      | FREDS            | *LOCAL.DB2.00... | 20028          | 20028         |
| 21                            | db2evm.exe       | DB2ADMIN         | *LOCAL.DB2.00... | 20028          | 10014         |
| 27                            | acmeacct.exe     | ALICET           | *LOCAL.DB2.00... | 10015          | 10015         |

Use the **Report data** pane to display and to work with the report data that is available to you within the Activity Monitor. The **Report data** pane displays the items that make up the contents of the report that is selected in the **Report** field.

The **Report data** pane also provides access to other Activity Monitor windows. From the Activity Monitor, you can drill down from the applications you are monitoring to the individual transactions or to the individual SQL statements that these applications are running.

## Report data pane toolbar



Use the toolbar located below the **Report data** pane to tailor the view of objects and information in the **Report data** pane to suit your needs.

---

## Monitoring scenarios

### Scenario: Identifying costly applications using snapshot administrative views

Recent increases in the workload on the ShopMart database have started hindering overall database performance. Jessie, the ShopMart DBA, is trying to identify the larger resource consumers in the daily workload using the following administrative views:

#### APPLICATION\_PERFORMANCE

This view helps Jessie identify applications that might be performing large table scans:

```
connect to shopmart;
select AGENT_ID, ROWS_SELECTED, ROWS_READ from APPLICATION_PERFORMANCE;
```

The value of ROWS\_SELECTED shows her how many rows are returned to an application and the value of ROWS\_READ shows her how many rows are accessed from the base tables. If the selectivity is low, the

application might be performing a table scan that could be avoided with the creation of an index. Jessie uses this view to identify potentially troublesome queries, and then she can investigate further by looking at the SQL to see if there are any ways to reduce the number of rows that are read in the execution of the query.

### **LONG\_RUNNING\_SQL**

Jessie uses the LONG\_RUNNING\_SQL administrative view to identify the longest running queries that are currently being executed:

```
connect to shopmart;
select ELAPSED_TIME_MIN, APPL_STATUS, AGENT_ID
 from long_running_sql order by ELAPSED_TIME_MIN desc
 fetch first 5 rows only;
```

Using this view, she can determine the length of time these queries have been running, and the status of these queries. If a query has been executing for a long time and is waiting on a lock, she can use the LOCKWAITS or LOCK\_HELD administrative views querying on a specific agent id to investigate further. The LONG\_RUNNING\_SQL view can also tell her the statement that is being executed, allowing her to identify potentially problematic SQL.

### **QUERY\_PREP\_COST**

Jessie uses the QUERY\_PREP\_COST to troubleshoot queries that have been identified as problematic. This view can tell her how frequent a query is run as well as the average execution time for each of these queries:

```
connect to shopmart;
select NUM_EXECUTIONS, AVERAGE_EXECUTION_TIME_S, PREP_TIME_PERCENT
 from QUERY_PREP_COST order by NUM_EXECUTIONS desc;
```

The value of PREP\_TIME\_PERCENT tells Jessie what percentage of the queries execution time is spent preparing the query. If the time it takes to compile and optimize a query is almost as long as it takes for the query to execute, Jessie might want to advise the owner of the query to change the optimization class used for the query. Lowering the optimization class might make the query complete optimization more rapidly and therefore return a result sooner. However, if a query takes a significant amount of time to prepare but is executed thousands of times (without being prepared again) then changing the optimization class might not benefit query performance.

### **TOP\_DYNAMIC\_SQL**

Jessie uses the TOP\_DYNAMIC\_SQL view to identify the most frequently executed, longest-running and most sort-intensive dynamic SQL statements. Having this information will allow Jessie to focus her SQL tuning efforts on the queries that represent some of the biggest resource consumers

To identify the most frequently run dynamic SQL statements, Jessie issues the following:

```
connect to shopmart;
select * from TOP_DYNAMIC_SQL order by NUM_EXECUTIONS desc
 fetch first 5 rows only;
```

This returns all of the details regarding the execution time, number of sorts performed, and the statement text for the five most frequent dynamic SQL statements.

To identify the dynamic SQL statements with the longest execution times, Jessie examines the queries with the top five values for AVERAGE\_EXECUTION\_TIME\_S:

```
connect to shopmart;
select * from TOP_DYNAMIC_SQL
order by AVERAGE_EXECUTION_TIME_S desc fetch first 5 rows only;
```

To look at the details of the most sort-intensive dynamic SQL statements, Jessie issues the following:

```
connect to shopmart;
select STMT_SORTS, SORTS_PER_EXECUTION, substr(STMT_TEXT,1,60) as STMT_TEXT
from TOP_DYNAMIC_SQL order by STMT_SORTS desc
fetch first 5 rows only;
```

## Scenario: Monitoring buffer pool efficiency using administrative views

John, a DBA, suspects that poor application performance in the SALES database is a result of buffer pools that function inefficiently. To investigate, he takes a look at the buffer pool hit ratio using the BP\_HITRATIO administrative view:

```
connect to SALES;
select BPNAME, TOTAL_HIT_RATIO from BP_HIT_RATIO;
```

John sees that the hit ratio for one of the buffer pools is very low, which means that too many pages are being read from disk instead of being read from the buffer pool.

He then decides to use the BP\_READ\_IO administrative view to see whether the prefetchers require tuning:

```
connect to SALES;
select BPNAME, PERCENT_SYNC_READS, UNUSED_ASYNC_READS_PERCENT from BP_READ_IO;
```

The value for PERCENT\_SYNC\_READS tells him the percentage of pages read synchronously without prefetching. A high number indicates that a high percentage of data is being read directly from disk, and might indicate that more prefetchers are required. The value of UNUSED\_ASYNC\_READS\_PERCENT tells him the percentage of pages read asynchronously from disk, but never accessed by a query. This might indicate that the prefetchers are overly aggressive in reading in data pages, resulting in unnecessary I/O.

Since both the values for PERCENT\_SYNC\_READS and UNUSED\_ASYNC\_READS\_PERCENT seem within the acceptable range, John uses the BP\_WRITE\_IO administrative view to investigate how well the page cleaners are working to clear space for incoming data pages:

```
connect to SALES;
select BPNAME, PERCENT_WRITES_ASYNC from BP_WRITE_IO;
```

The value of PERCENT\_WRITES\_ASYNC tells John what percentage of physical write requests that were performed asynchronously. If this number is high, it might mean that the page cleaners are working well to clear space in the buffer pool ahead of incoming requests for new data pages. If this number is low, then a higher number of physical writes are being performed by database agents while an application waits for data a data page to be read into the buffer pool.

John sees that the value of PERCENT\_WRITES\_ASYNC is very low at 25 percent, so he decides to configure more page cleaners for the SALES database to increase

the rate of asynchronous writes. After increasing the number of page cleaners, he can use the buffer pool administrative views again to see the effects of his tuning.

---

## Setting up an activity monitor

To monitor application performance and concurrency, resource consumption, and SQL statement usage of a database or database partition, you can set up an activity monitor. The Activity Monitor provides a set of predefined reports based on a specific subset of monitor data. It can also provide recommendations to assist you in diagnosing the cause of database performance problems, and to tune queries so that your use of database resources is optimized.

To use the Activity Monitor:

- Your server must have DB2 UDB Version 8.2 or later
- You must have DBADM authority

Open the Set Up Activity Monitor wizard.

- From the Control Center, expand the object tree until you find the instance or the database for which you want to set up an activity monitor. Right-click on the object and select Set Up Activity Monitor from the pop-up menu.
- From the command line, type the following command: `db2am`.

Detailed information is provided through the contextual help facility within the Control Center.

---

## Progress monitoring of the runtime rollback process

Progress monitoring of runtime rollback provides progress information of rollback events using application snapshots. Rollback events are of two types:

### Unit of work rollback

Includes explicit (user invoked) and implicit (forced) rollback of the entire transaction.

### Savepoint rollback

Includes statement and application level savepoints. Nested savepoints are considered a single unit, using the outermost savepoint.

The information provided is the start time of the rollback event, the total work to be done, and completed work. The work metric is bytes.

Total Work units is the range in the log stream that needs to be rolled back for the transaction or savepoint.

Completed Work units shows the relative position in the log stream that has been rolled back.

Updates to Completed Work are made after every log record is processed. Updates are not performed evenly because log records vary in size.

## Sample output from GET SNAPSHOT FOR ALL APPLICATIONS command

Application Snapshot

```
Application handle = 6
Application status = Rollback Active
```

```
Start Time = 02/20/2004 12:49:27.713720
Completed Work = 1024000 bytes
Total Work = 4084000 bytes
```

#### Application Snapshot

```
Application handle = 10
Application status = Rollback to Savepoint
 Start Time = 02/20/2004 12:49:32.832410
 Completed Work = 102400 bytes
 Total Work = 2048000 bytes
```

**Note:** If rollback is not active during a snapshot, then rollback elements will not be displayed.

---

## Using snapshot monitor data to monitor the reorganization of a partitioned table

The following information describes some of the most useful methods of monitoring the global status of a table reorganization.

There is no separate data group indicating the overall table reorganization status for a partitioned table. A partitioned table uses a data organization scheme in which table data is divided across multiple storage objects, called data partitions or ranges, according to values in one or more table partitioning key columns of the table. However, you can deduce the global status of a table reorganization from the values of elements in the individual data partition data group being reorganized. The following information describes some of the most useful methods of monitoring the global status of a table reorganization.

### Determining the number of data partitions being reorganized

You can determine the total number of data partitions being reorganized on a table by counting the number of monitor data blocks for table data that have the same table name and schema name. This value indicates the number of data partitions on which reorganization has started. Examples 1 and 2 indicate that three data partitions are being reorganized.

### Identifying the data partition being reorganized

You can deduce the current data partition being reorganized from the phase start time (`reorg_phase_start`). During the SORT/BUILD/REPLACE phase, the monitor data corresponding to the data partition that is being reorganized shows the most recent phase start time. During the INDEX\_RECREATE phase, the phase start time is the same for all the data partitions. In Examples 1 and 2, the INDEX\_RECREATE phase is indicated, so the start time is the same for all the data partitions.

### Identifying an index rebuild requirement

You can determine if an index rebuild is required by obtaining the value of the maximum reorganize phase element (`reorg_max_phase`), corresponding to any one of the data partitions being reorganized. If `reorg_max_phase` has a value of 3 or 4, then an Index Rebuild is required. Examples 1 and 2 report a `reorg_max_phase` value of 3, indicating an index rebuild is required.

The following sample output is from a three-node server that contains a table with three data partitions:

```

CREATE TABLE sales (c1 INT, c2 INT, c3 INT)
PARTITION BY RANGE (c1)
(PART P1 STARTING FROM (1) ENDING AT (10) IN parttbs,
PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
DISTRIBUTE BY (c2)

```

Statement executed:

```

REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS

```

*Example 1:*

```

GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL

```

The output is modified to include table information for the relevant table only.

#### Table Snapshot

```

First database connect timestamp = 06/28/2005 13:46:43.061690
Last reset timestamp = 06/28/2005 13:46:47.440046
Snapshot timestamp = 06/28/2005 13:46:50.964033
Database name = DPARTDB
Database path = /work/sales/NODE0000/SQL00001/
Input database alias = DPARTDB
Number of accessed tables = 5

```

#### Table List

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 0
Data Object Pages = 3
Rows Read = 12
Rows Written = 1
Overflows = 0
Page Reorgs = 0
Table Reorg Information:
Node number = 0
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:49.816883
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.362918
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.821244

```

#### Table Reorg Information:

```

Node number = 1
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0

```

```

Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:49.822701
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.420741
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.899543

```

Table Reorg Information:

```

Node number = 2
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:49.814813
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 1
Data Object Pages = 3
Rows Read = 8
Rows Written = 1
Overflows = 0
Page Reorgs = 0
Table Reorg Information:
 Node number = 0
 Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
 Reorg Index = 0
 Reorg Tablespace = 3
 Long Temp space ID = 3
 Start Time = 06/28/2005 13:46:50.014617
 Reorg Phase = 3 - Index Recreate
 Max Phase = 3
 Phase Start Time = 06/28/2005 13:46:50.362918
 Status = Completed
 Current Counter = 0
 Max Counter = 0
 Completion = 0
 End Time = 06/28/2005 13:46:50.821244

```

Table Reorg Information:

```

Node number = 1

```



```

Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.026278
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.420741
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.899543

```

Table Reorg Information:

```

Node number = 2
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.006392
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 2
Data Object Pages = 3
Rows Read = 4
Rows Written = 1
Overflows = 0
Page Reorgs = 0
Table Reorg Information:
Node number = 0
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.199971
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.362918
Status = Completed
Current Counter = 0

```

```

Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.821244

```

Table Reorg Information:

```

Node number = 1
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.223742
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.420741
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.899543

```

Table Reorg Information:

```

Node number = 2
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.179922
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

*Example 2:*

**GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2**

The output is modified to include table information for the relevant table only.

Table Snapshot

```

First database connect timestamp = 06/28/2005 13:46:43.617833
Last reset timestamp =
Snapshot timestamp = 06/28/2005 13:46:51.016787
Database name = DPARTDB
Database path = /work/sales/NODE0000/SQL00001/
Input database alias = DPARTDB
Number of accessed tables = 3

```

Table List

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 0

```

```

Data Object Pages = 1
Rows Read = 0
Rows Written = 0
Overflows = 0
Page Reorgs = 0
Table Reorg Information:
 Node number = 2
 Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
 Reorg Index = 0
 Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:49.814813
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 1
Data Object Pages = 1
Rows Read = 0
Rows Written = 0
Overflows = 0
Page Reorgs = 0
Table Reorg Information:
 Node number = 2
 Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
 Reorg Index = 0
 Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.006392
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

```

Table Schema = NEWTON
Table Name = SALES
Table Type = User
Data Partition Id = 2
Data Object Pages = 1
Rows Read = 4
Rows Written = 1
Overflows = 0
Page Reorgs = 0

```

```

Table Reorg Information:
Node number = 2
Reorg Type =
 Reclaiming
 Table Reorg
 Allow No Access
 Recluster Via Table Scan
 Reorg Data Only
Reorg Index = 0
Reorg Tablespace = 3
Long Temp space ID = 3
Start Time = 06/28/2005 13:46:50.179922
Reorg Phase = 3 - Index Recreate
Max Phase = 3
Phase Start Time = 06/28/2005 13:46:50.344277
Status = Completed
Current Counter = 0
Max Counter = 0
Completion = 0
End Time = 06/28/2005 13:46:50.803619

```

Example 3:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

The output is modified to include a subset of table information for the relevant table only.

| ... | TBSP_NAME | TABNAME | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS | ... |
|-----|-----------|---------|------------------|-----------|-------------|-----|
| ... | PARTTBS   | SALES   | ROW_LOCK         | X         | GRNT        | ... |
| ... | -         | SALES   | TABLE_LOCK       | IX        | GRNT        | ... |
| ... | PARTTBS   | SALES   | TABLE_PART_LOCK  | IX        | GRNT        | ... |
| ... | PARTTBS   | SALES   | ROW_LOCK         | X         | GRNT        | ... |
| ... | -         | SALES   | TABLE_LOCK       | IX        | GRNT        | ... |
| ... | PARTTBS   | SALES   | TABLE_PART_LOCK  | IX        | GRNT        | ... |
| ... | PARTTBS   | SALES   | ROW_LOCK         | X         | GRNT        | ... |
| ... | -         | SALES   | TABLE_LOCK       | IX        | GRNT        | ... |
| ... | PARTTBS   | SALES   | TABLE_PART_LOCK  | IX        | GRNT        | ... |

9 record(s) selected.

Output from this query (continued).

| ... | LOCK_ESCALATION | LOCK_ATTRIBUTES | DATA_PARTITION_ID | DBPARTITIONNUM |
|-----|-----------------|-----------------|-------------------|----------------|
| ... | 0               | INSERT          | 2                 | 2              |
| ... | 0               | NONE            | -                 | 2              |
| ... | 0               | NONE            | 2                 | 2              |
| ... | 0               | INSERT          | 0                 | 0              |
| ... | 0               | NONE            | -                 | 0              |
| ... | 0               | NONE            | 0                 | 0              |
| ... | 0               | INSERT          | 1                 | 1              |
| ... | 0               | NONE            | -                 | 1              |
| ... | 0               | NONE            | 1                 | 1              |

Example 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

The output is modified to include a subset of table information for the relevant table only.

| ... | TABSCHEMA | TABNAME | TAB_FILE_ID | TAB_TYPE   | DATA_OBJECT_PAGES | ROWS_WRITTEN | ... |
|-----|-----------|---------|-------------|------------|-------------------|--------------|-----|
| ... | NEWTON    | SALES   | 2           | USER_TABLE | 1                 | 1            | ... |

```

... NEWTON SALES 4 USER_TABLE 1 1 ...
... NEWTON SALES 3 USER_TABLE 1 1 ...

```

3 record(s) selected.

Output from this query (continued).

```

... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... -----
... 0 0 0 3 0
... 0 0 2 3 2
... 0 0 1 3 1

```

Example 5:

```

SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;

```

The output is modified to include a subset of table information for the relevant table only.

```

REORG_PHASE REORG_MAX_PHASE REORG_TYPE

INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...
INDEX_RECREATE 3 RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY ...

```

9 record(s) selected.

Output from this query (continued).

```

... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
... -----
... COMPLETED 3 2 0
... COMPLETED 3 2 1
... COMPLETED 3 2 2
... COMPLETED 3 1 0
... COMPLETED 3 1 1
... COMPLETED 3 1 2
... COMPLETED 3 0 0
... COMPLETED 3 0 1
... COMPLETED 3 0 2

```

---

## Inactive statement tracking for DEADLOCK WITH DETAILS HISTORY event monitors

When running a deadlock event monitor that tracks all statements (and optionally data values) it becomes possible for the system monitor heap to be exhausted by a single application that includes a very high number of statements in a unit of work. It is also possible for the monitor heap to become exhausted if there are a large number of applications executing concurrently.

To mitigate the amount of space that is consumed, inactive statements are written out to the event monitor by an application when the number of inactive statements for that application reaches a certain threshold. After being written to the event monitor, the memory consumed by these inactive statements will be released. Also, if at any time an application is unable to acquire memory from the system monitor heap, that application writes out all of its current inactive statements to the event

monitor before trying to acquire the memory again. If the second attempt fails, a message is logged, and the statement history list is truncated for the UOW the application is processing.

The default limit on the number of inactive statements to be kept by any one application is 250. This default value can be overridden using the registry variable `DB2_MAX_INACT_STMTS` to specify a different value. 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.

Whenever inactive statements are written out to the event monitor, a message appears in the `db2diag.log` file indicating that this has occurred. Whenever the limit for inactive statements has been exceeded, a message appears in the `db2diag.log` file indicating that this has occurred.

Because an application can now record its statement history entries outside of the context of a deadlock (when one of the thresholds mentioned above are reached), we need a mechanism to associate these entries to the list of statements recorded at the time of a deadlock for analysis. To do this, users can look for statement history entries where:

- `deadlock_id= 0`
- `participant_no = 0`
- `invocation_id= invocation id of the deadlock`
- `application_id= application identifier of the application that participated in the deadlock`

In the case of a write to table event monitor, the number of `evmon_activates` also needs to be checked.

**Notes:**

- For SQL statements compiled using the `REOPT ALWAYS` bind option, there will be no reopt compilation or statement execution data values provided in the deadlock event information.
- At coordinator nodes, when inactive statements are written to the event monitor due to the conditions described in the previous section, the sequence value of all records written will be changed to reflect the current unit of work in process. This is done to help reconcile this data with any data generated later by a deadlock in the same unit of work since all the relevant data can be gathered by searching for the sequence number and application ID information for those records with `deadlock_id` of 0. This change does mean that the unit of work information is not available for statements started in a preceding unit of work but still active in the current work, as the sequence number will be overwritten by the current unit of work identifier. This behavior does not occur at remote nodes (that is, the original unit of work information is not overwritten) and so care must be taken when trying to reconcile deadlock event records with any records written out prior to the deadlock as the sequence numbers might differ if there are active cursors with hold from previous units of work involved.

---

## Chapter 6. Working with the Memory Visualizer

The Memory Visualizer helps database administrators to monitor the memory-related performance of an instance and all of its databases. You can view a live, visual display of the memory utilization of memory components organized in a hierarchical tree.

To view memory performance and usage plots and to update the configuration parameters in the Memory Visualizer, you must have SYSADM authority.

You can use the Memory Visualizer to troubleshoot performance problems. You can change the configuration parameter settings for a memory component and assess the effect that the changes have. Configuration parameters affect memory usage in DB2 because memory is allocated as it is required. If you set the value of a configuration parameter above or below its acceptable range, an error message will display. Changing the configuration parameter takes immediate effect within the Memory Visualizer, and the new value is integrated in the next refresh cycle.

- To view memory performance using the Memory Visualizer:
  1. Open the Memory Visualizer from the Windows **Start** menu, by clicking **Programs** → **IBM DB2** → **Monitoring** → **Tools** → **Memory Visualizer**. The Memory Visualizer instance selection window opens. Select an instance from the **Instance name** field and click **OK**.
  2. Expand the instance object tree until you display the databases and their associated memory components in the hierarchical tree. Values for the memory pools display in the Memory Visualizer window.
  3. To display a plotted graph of a memory component, use one of the following methods:
    - Select a component in the hierarchical tree and click the **Show Plot** check box in the Memory Visualizer window.
    - Right-click the selected memory component to display a pop-up menu and select **Show Plot**.
    - Select a component in the hierarchical tree and select the **Show Plot** option from the Selected menu on the tool bar. The plotted data for each memory component appears in the Memory Usage Plot.
    - To view data from another memory component, select it from the hierarchical tree and click the **Show Plot** check box. The plotted data for the component appears in the Memory Usage Plot along with other components.

The graph displays data collected for memory components over time. Each component is represented by a color and shape which also displays in the **Plot Legend** field in the Memory Visualizer window. The shape is repeated at intervals. A label identifies the component in the graph plot.

The time that the performance data was captured is displayed below the graph. You can change the time interval for the graph.

**Note:** When a new memory component is added to the plot, it does not replace the memory components that were previously added.

Horizontal and vertical scroll bars offer different views of the plotted data.

- Use the horizontal scroll bar, located at the bottom of the graph, to view historical data of the memory component over a selected time period. Point to and drag the slider bar along the base of the graph.
- Use the vertical scroll bar, located at the right of the graph, to view the memory utilization of the selected component. Point to and drag the slider to change the view.

When the memory utilization reaches a new high, the maximum value of the vertical scroll bar is updated to reflect the new value. You can set the minimum value of the vertical scroll bar to a value other than 0 to view a different range of pool utilization values.

- You can load data from a Memory Visualizer data file into a new Memory Visualizer window. This data can be used to compare the performance of an instance and all of its databases against historical data. To load data from a Memory Visualizer data file, select **Open** from the Memory Visualizer menu, and then from the Open Dialog select a data file with extension \*.mdf.
- Use the **Time Unit** field to change the time interval on the Memory Usage Plot window. The default time interval for the graph data is minutes. You can select intervals of minutes, hours, or days. When selected, a new time interval displays in the horizontal range of the graph and changes the incremental movement of the horizontal scroll bar.
- To remove the plotted graph of a memory component from the Memory Usage Plot, either select a component in the hierarchical tree and clear the **Show Plot** check box in the Memory Visualizer window, or right-click the selected memory component to display a pop-up menu and deselect **Show Plot**. The plotted data for the component is removed from the Memory Usage Plot window. The colored shape, which represented the component, no longer displays in the **Plot Legend** field in the Memory Visualizer window.
- To help you to track and create a history of memory performance, you can save memory performance data, including plotted graphs, while the Memory Visualizer is running. To save memory performance data, select **Save** or **Save As** from the Memory Visualizer menu, and then select a location for the file and a filename with extension .mdf.
- To change the configuration parameter settings for a memory component:
  1. Expand the memory pool that you want so that you can see its configuration parameters listed in the hierarchical tree.
  2. Click a component to select it and click the number in the **Parameter Value** column. A text box displays the current value for the component. Type a new number in the text box and press **Enter**. The new value displays next to the original value in the **Parameter Value** column until the configuration parameter is updated possibly in the next refresh cycle. You can also right-click the value in the **Parameter Value** column for the selected component to display the pop-up menu. Click outside of the column to complete the change. The new value for the memory component displays next to the original value in the **Parameter Value** column. If you select to view a graph of memory performance, you will see the new value in the graph plot view. While this change takes place immediately in the Memory Visualizer, there is a delay in updating the change you made to the configuration parameter within DB2. You can reset the value of the configuration parameter using the **Reset to Default** option in the pop-up menu.



---

## Memory Visualizer overview

Use the Memory Visualizer to monitor the memory-related performance of an instance and all of its databases.

Open the Memory Visualizer and select a memory component or multiple components in the hierarchical tree to display values for the amount of memory allocated to the component and the current memory usage in the Memory Visualizer window. The Memory Visualizer window displays two views of data: a tree view and a historical view. A series of columns show percentage threshold values for upper and lower alarms and warnings. The columns also display real time memory utilization.

**Note:** The Memory Visualizer is available to provide memory performance data for instances that are Version 8.1 and later.

The following list categorizes some of the key tasks that you can do with the Memory Visualizer:

- View or hide data in various columns on the memory utilization of selected components for a DB2 instance and its databases.
- View a graph of memory performance data.
- Change settings for individual memory components by updating configuration parameters.
- Load performance data from a file into a Memory Visualizer window.
- Save the memory performance data.

The Memory Visualizer interface has the following elements that help you monitor the memory-related performance of an instance and all of its databases.

### The Memory Visualizer window

The columns in the Memory Visualizer window display values for the performance of memory components. The following information is shown:

#### Plot Legend

The checked memory components or configuration parameters shown in the Memory Usage Plot. A specific shape that occurs at regular intervals in the plotted graph identifies each component or parameter.

#### Utilization

The size of the memory that is allocated to, and utilized by, the database object. Includes a graphical bar showing the utilization and configured allocation. The length of the bar is fixed and the filled portion indicates utilization as a percentage.

#### Parameter Value

The current value of a configuration parameter.

#### Upper Alarm (%) Threshold

The threshold value that generates an upper alarm. The default value is 98%.

#### Upper Warning (%) Threshold

The threshold value that generates an upper warning. The default value is 90%.

**Lower Alarm (%) Threshold**

The threshold value that generates a lower alarm. The default value is 2%.

**Lower Warning (%) Threshold**

The threshold value that generates a lower warning. The default value is 10%.

**Graphical Usage Bars**

The graphical usage bars in the Memory Visualizer window are visual cues of memory utilization. The bars can assist you in determining how much memory is being used by selected memory components and the potential effect that the usage can have on the system. The Memory Visualizer also displays a percentage value that corresponds to the usage. These two indicators can help you to determine whether you need to change the configuration parameter setting for the component or take another appropriate action.

**Memory Components**

The Database Manager uses different types of memory on a system, namely Database manager shared memory, Database global memory, Application global memory, Agent /Application shared memory, and Agent private memory. These types of memory are the high level memory components that the Memory Visualizer uses in its expanding hierarchical tree organization.

Underlying each high-level memory component are other components that determine how the memory is allocated and deallocated. For example, when the database manager starts, a database is activated, an application connects to a database, or when an agent is assigned to work for an application, memory is allocated and deallocated. The Memory Visualizer uses these leaf-level memory components to display how memory is allocated and used in a DB2 instance. For more information on how DB2 uses memory, see the Administration Guide.

**Hierarchical Tree Organization**

The Memory Visualizer uses a hierarchical tree organization to help you to display and browse the memory components in DB2. The hierarchical tree allows you to expand and view information on individual memory components through columns, graphical displays, and graphs.

The tree view comprises four major types of memory items:

**DB2 Instance**

The instance that is currently running on the system

**Databases**

The databases defined on the instance





**High-level memory components**

Logical groupings for leaf-level memory components. These groups are: Database manager shared memory, Database global memory, Agent private memory, Agent /Application shared memory

**Leaf-level memory components**

The memory components that display in the Memory Visualizer window such as buffer pools, sort heaps, database heap, and lock list.

Icons in the tree view represent each memory tree item:

- Instance: 
- Database: 
- High-level memory groupings: 
- Leaf-level memory components: 

If the memory utilization for a tree items exceeds a threshold value, a colored indicator overlays the icon. The yellow color indicates a warning condition. The red color indicates an alarm condition.

The historical view displays data for memory components selected in the tree view. The data includes values for memory allocated and utilized, plotted graphs, as well as changes made to the configuration parameters while the Memory Visualizer is running. The data is saved for a specific period within the Memory Visualizer. You can save memory performance data to a Memory Visualizer data file for tracking, comparing with other data, or troubleshooting.

### The Memory Usage Graph

The memory usage graph displays plotted data for selected memory components in the Memory Usage Plot. Each component in the graph is identified by a specific color, which also displays in the Plot Legend column in the Memory Visualizer window. The graph also displays changes made to the configuration parameters settings. The original value of the configuration parameter and the new value setting appear in the graph, in addition to the time that the change was requested. They become part of the history view that you can use in assessing memory performance.

For further details, see Chapter 6, “Working with the Memory Visualizer,” on page 93.



---

## Chapter 7. Monitoring database systems (Windows)

---

### 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.

---

## Windows performance monitor introduction

When working with DB2 database manager for Windows, there are tools that can be used to monitor performance:

- **DB2 Performance Expert**

DB2 Performance Expert for Multiplatforms, Version 1.1 consolidates, reports, analyzes and recommends self-managing and resource tuning changes based on DB2 database performance-related information.

- **DB2 Health Center**

The functions of the Health Center provide you with different methods to work with performance-related information. These functions somewhat replace the performance monitor capability of the Control Center.

- **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

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

If your DB2 for Windows workstation is networked to other Windows computers, you can use the feature described in this section.

In order 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.) 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

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

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

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. System monitor elements**



---

## Chapter 8. Logical data groups

---

### 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 14. Snapshot Monitor Interface Mappings to Logical Data Groups*

| 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           |
| SQLMA_DB2              | get snapshot for<br>dbm                                             | SNAPDBM                 | db2                     |
|                        |                                                                     | SNAPFCM                 | fcm                     |
|                        |                                                                     | SNAPFCMPART             | fcm_node                |
|                        |                                                                     | SNAPUTIL                | utility_info            |
|                        |                                                                     | SNAPUTIL_PROGRESS       | progress, progress_info |
|                        |                                                                     | SNAPDBM_MEMORY_POOL     | memory_pool             |
|                        | get dbm monitor<br>switches                                         | SNAPSWITCHES            | switch_list             |

Table 14. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

| API request type       | CLP command                                             | SQL administrative view | Logical data groups           |
|------------------------|---------------------------------------------------------|-------------------------|-------------------------------|
| SQLMA_DBASE            | get snapshot for database on <i>dbname</i>              | SNAPDB                  | dbase                         |
|                        |                                                         | SNAPDETAILLOG           | detail_log                    |
|                        |                                                         | SNAPSTORAGE_PATHS       | db_storage_group              |
|                        |                                                         |                         | rollforward                   |
|                        |                                                         | SNAPTbsp                | tablespace                    |
|                        | SNAPDB_MEMORY_POOL                                      | memory_pool             |                               |
| SQLMA_DBASE_ALL        | get snapshot for all databases                          | SNAPDB                  | dbase                         |
|                        |                                                         | SNAPSTORAGE_PATHS       | db_storage_group              |
|                        |                                                         |                         | rollforward                   |
|                        |                                                         | 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                          |
|                        |                                                         | SNAPAGENT_MEMORY_POOL   | memory_pool                   |
| SQLMA_AGENT_ID         | get snapshot for application agentid <i>appl-handle</i> | SNAPAGENT               | appl                          |
|                        |                                                         | SNAPAGENT               | agent                         |
|                        |                                                         | SNAPAPPL_INFO           | appl_info                     |
|                        |                                                         | SNAPLOCKWAIT            | lock_wait                     |
|                        |                                                         | SNAPSTMT                | stmt                          |
|                        |                                                         | SNAPSUBSECTION          | subsection                    |
|                        | SNAPAGENT_MEMORY_POOL                                   | memory_pool             |                               |

Table 14. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

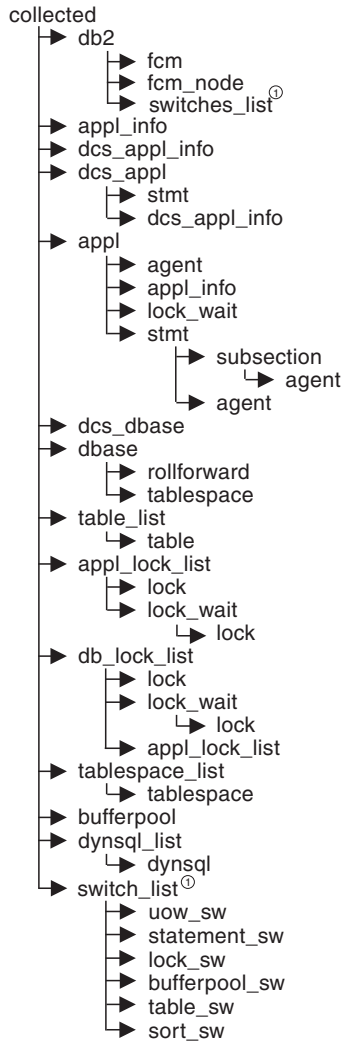
| API request type          | CLP command                                                       | SQL administrative view          | Logical data groups                                   |
|---------------------------|-------------------------------------------------------------------|----------------------------------|-------------------------------------------------------|
| SQLMA_DBASE_APPLS         | get snapshot for applications on <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                                            |
|                           |                                                                   | SNAPAGENT_MEMORY_POOL            | memory_pool                                           |
| SQLMA_DCS_APPL            | get snapshot for dcs application applid <i>appl-id</i>            |                                  | dcx_appl, dcs_stmt, dcs_appl_info, stmt_transmissions |
| SQLMA_DCS_APPL_ALL        | get snapshot for all dcs applications                             |                                  | dcx_appl, dcs_stmt, dcs_appl_info, stmt_transmissions |
| SQLMA_DCS_APPL_HANDLE     | get snapshot for dcs application agentid <i>appl-handle</i>       |                                  | dcx_appl, dcs_stmt, dcs_appl_info, stmt_transmissions |
| SQLMA_DCS_DBASE_APPLS     | get snapshot for dcs applications on <i>dbname</i>                |                                  | dcx_appl, dcs_stmt, dcs_appl_info, stmt_transmissions |
| SQLMA_DBASE_APPLS_REMOTE  | get snapshot for remote applications on <i>dbname</i>             |                                  | dbase_appl                                            |
| SQLMA_APPL_REMOTE_ALL     | get snapshot for all remote applications                          |                                  | dbase_appl                                            |
| 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                       |

Table 14. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

| API request type        | CLP command                                   | SQL administrative view | Logical data groups                       |
|-------------------------|-----------------------------------------------|-------------------------|-------------------------------------------|
| 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 5. 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 table lists the logical data groupings and monitor elements that can be returned by snapshot monitoring.

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements

| Snapshot logical data groups | Monitor element                                                                                                           |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| agent                        | “agent_pid - Engine dispatchable unit (EDU) monitor element” on page 204<br>“lock_timeout_val - Lock timeout” on page 315 |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                             |
|------------------------------|-----------------------------------------------------------------------------|
| appl                         | "acc_curs_blk - Accepted Block Cursor Requests" on page 362                 |
|                              | "agent_sys_cpu_time - System CPU Time used by Agent" on page 400            |
|                              | "agent_usr_cpu_time - User CPU Time used by Agent" on page 400              |
|                              | "agents_stolen - Stolen Agents" on page 213                                 |
|                              | "appl_con_time - Connection Request Start Timestamp" on page 199            |
|                              | "appl_idle_time - Application Idle Time" on page 204                        |
|                              | "appl_priority - Application Agent Priority" on page 195                    |
|                              | "appl_priority_type - Application Priority Type" on page 196                |
|                              | "associated_agents_top - Maximum Number of Associated Agents" on page 214   |
|                              | "authority_lvl - User Authorization Level" on page 196 (deprecated)         |
|                              | "authority_bitmap - User Authorization Level monitor element" on page 197   |
|                              | "binds_precompiles - Binds/Precompiles Attempted" on page 372               |
|                              | "cat_cache_inserts - Catalog Cache Inserts" on page 273                     |
|                              | "cat_cache_lookups - Catalog Cache Lookups" on page 272                     |
|                              | "cat_cache_overflows - Catalog Cache Overflows" on page 274                 |
|                              | "commit_sql_stmts - Commit Statements Attempted" on page 365                |
|                              | "conn_complete_time - Connection Request Completion Timestamp" on page 200  |
|                              | "ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 368 |
|                              | "deadlocks - Deadlocks Detected" on page 298                                |
|                              | "direct_read_reqs - Direct Read Requests" on page 270                       |
|                              | "direct_read_time - Direct Read Time" on page 271                           |
|                              | "direct_reads - Direct Reads From Database" on page 268                     |
|                              | "direct_write_reqs - Direct Write Requests" on page 270                     |
|                              | "direct_write_time - Direct Write Time" on page 271                         |
|                              | "direct_writes - Direct Writes to Database" on page 269                     |
|                              | "dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 364          |
|                              | "failed_sql_stmts - Failed Statement Operations" on page 364                |
|                              | "hash_join_overflows - Hash Join Overflows" on page 229                     |
|                              | "hash_join_small_overflows - Hash Join Small Overflows" on page 230         |
|                              | "inbound_comm_address - Inbound Communication Address" on page 437          |
|                              | "int_auto_rebinds - Internal Automatic Rebinds" on page 369                 |
|                              | "int_commits - Internal Commits" on page 369                                |
|                              | "int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 371   |
|                              | "int_rollbacks - Internal Rollbacks" on page 370                            |
|                              | "int_rows_deleted - Internal Rows Deleted" on page 349                      |
|                              | "int_rows_inserted - Internal Rows Inserted" on page 351                    |
|                              | "int_rows_updated - Internal Rows Updated" on page 350                      |
|                              | "last_reset - Last Reset Timestamp" on page 405                             |
|                              | "lock_escalation - Lock Escalation" on page 306                             |
|                              | "lock_timeouts - Number of Lock Timeouts" on page 305                       |
|                              | "lock_timeout_val - Lock timeout" on page 315                               |
|                              | "lock_wait_time - Time Waited On Locks" on page 313                         |
|                              | "lock_waits - Lock Waits" on page 312                                       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| appl (continued)             | <p>"locks_held - Locks Held" on page 297</p> <p>"locks_waiting - Current Agents Waiting On Locks" on page 314</p> <p>"num_agents - Number of Agents Working on a Statement" on page 399</p> <p>"olap_func_overflows - OLAP Function Overflows monitor element" on page 231</p> <p>"open_loc_curs - Open Local Cursors" on page 362</p> <p>"open_loc_curs_blk - Open Local Cursors with Blocking" on page 362</p> <p>"open_rem_curs - Open Remote Cursors" on page 360</p> <p>"open_rem_curs_blk - Open Remote Cursors with Blocking" on page 361</p> <p>"pkg_cache_inserts - Package Cache Inserts" on page 277</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 276</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 240</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 245</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 250</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244</p> <p>"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247</p> <p>"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 251</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes" on page 250</p> <p>"prefetch_wait_time - Time Waited for Prefetch" on page 264</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 201</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows" on page 282</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 284</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 283</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size" on page 282</p> <p>"rej_curs_blk - Rejected Block Cursor Requests" on page 361</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 366</p> <p>"rows_deleted - Rows Deleted" on page 345</p> <p>"rows_inserted - Rows Inserted" on page 345</p> <p>"rows_read - Rows Read" on page 348</p> <p>"rows_selected - Rows Selected" on page 346</p> <p>"rows_updated - Rows Updated" on page 346</p> <p>"rows_written - Rows Written" on page 347</p> <p>"select_sql_stmts - Select SQL Statements Executed" on page 367</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows" on page 280</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 281</p> |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                |
|------------------------------|--------------------------------------------------------------------------------|
| appl (continued)             | "shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 280 |
|                              | "shr_workspace_size_top - Maximum Shared Workspace Size" on page 279           |
|                              | "sort_overflows - Sort Overflows" on page 224                                  |
|                              | "sql_reqs_since_commit - SQL Requests Since Last Commit" on page 372           |
|                              | "static_sql_stmts - Static SQL Statements Attempted" on page 363               |
|                              | "total_hash_joins - Total Hash Joins" on page 227                              |
|                              | "total_hash_loops - Total Hash Loops" on page 229                              |
|                              | "total_olap_funcs - Total OLAP Functions monitor element" on page 230          |
|                              | "total_sort_time - Total Sort Time" on page 223                                |
|                              | "total_sorts - Total Sorts" on page 222                                        |
|                              | "uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 367     |
|                              | "unread_prefetch_pages - Unread Prefetch Pages" on page 264                    |
|                              | "uow_comp_status - Unit of Work Completion Status" on page 203                 |
|                              | "uow_elapsed_time - Most Recent Unit of Work Elapsed Time" on page 202         |
|                              | "uow_lock_wait_time - Total Time Unit of Work Waited on Locks" on page 314     |
|                              | "uow_log_space_used - Unit of Work Log Space Used" on page 289                 |
|                              | "uow_start_time - Unit of Work Start Timestamp" on page 201                    |
|                              | "uow_stop_time - Unit of Work Stop Timestamp" on page 202                      |
|                              | "x_lock_escals - Exclusive Lock Escalations" on page 300                       |
|                              | "xquery_stmts - XQuery Statements Attempted" on page 373                       |
| appl_id_info                 | "agent_id - Application Handle (agent ID)" on page 180                         |
|                              | "appl_id - Application ID" on page 185                                         |
|                              | "appl_name - Application Name" on page 185                                     |
|                              | "appl_status - Application Status" on page 181                                 |
|                              | "auth_id - Authorization ID" on page 188                                       |
|                              | "client_db_alias - Database Alias Used by Application" on page 190             |
|                              | "client_prdid - Client Product/Version ID" on page 189                         |
|                              | "codepage_id - ID of Code Page Used by Application" on page 183                |
|                              | "db_name - Database Name" on page 172                                          |
|                              | "db_path - Database Path" on page 173                                          |
|                              | "input_db_alias - Input Database Alias" on page 405                            |
|                              | "sequence_no - Sequence number monitor element" on page 187                    |
|                              | "status_change_time - Application Status Change Time" on page 183              |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                           |
|------------------------------|---------------------------------------------------------------------------|
| appl_info                    | "agent_id - Application Handle (agent ID)" on page 180                    |
|                              | "appl_id - Application ID" on page 185                                    |
|                              | "appl_name - Application Name" on page 185                                |
|                              | "appl_section_inserts - Section Inserts monitor element" on page 285      |
|                              | "appl_section_lookups - Section Lookups" on page 285                      |
|                              | "appl_status - Application Status" on page 181                            |
|                              | "auth_id - Authorization ID" on page 188                                  |
|                              | "authority_lvl - User Authorization Level" on page 196 (deprecated)       |
|                              | "authority_bitmap - User Authorization Level monitor element" on page 197 |
|                              | "client_db_alias - Database Alias Used by Application" on page 190        |
|                              | "client_pid - Client Process ID" on page 193                              |
|                              | "client_platform - Client Operating Platform" on page 194                 |
|                              | "client_prdid - Client Product/Version ID" on page 189                    |
|                              | "client_protocol - Client Communication Protocol" on page 194             |
|                              | "codepage_id - ID of Code Page Used by Application" on page 183           |
|                              | "coord_agent_pid - Coordinator Agent monitor element" on page 204         |
|                              | "coord_node - Coordinating Node" on page 199                              |
|                              | "corr_token - DRDA Correlation Token" on page 193                         |
|                              | "db_name - Database Name" on page 172                                     |
|                              | "db_path - Database Path" on page 173                                     |
|                              | "execution_id - User Login ID" on page 192                                |
|                              | "input_db_alias - Input Database Alias" on page 405                       |
|                              | "is_system_appl - Is System Application monitor element" on page 191      |
|                              | "num_assoc_agents - Number of Associated Agents" on page 215              |
|                              | "sequence_no - Sequence number monitor element" on page 187               |
|                              | "status_change_time - Application Status Change Time" on page 183         |
|                              | "territory_code - Database Territory Code" on page 195                    |
|                              | "tpmon_acc_str - TP Monitor Client Accounting String" on page 460         |
|                              | "tpmon_client_app - TP Monitor Client Application Name" on page 459       |
|                              | "tpmon_client_userid - TP Monitor Client User ID" on page 459             |
|                              | "tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 459     |
|                              | "workload_id - Workload ID monitor element" on page 500                   |
|                              | "session_auth_id - Session Authorization ID" on page 189                  |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                        | Monitor element                                                    |
|---------------------------------------------------------------------|--------------------------------------------------------------------|
| appl_lock_list                                                      | "agent_id - Application Handle (agent ID)" on page 180             |
|                                                                     | "appl_id - Application ID" on page 185                             |
|                                                                     | "appl_name - Application Name" on page 185                         |
|                                                                     | "appl_status - Application Status" on page 181                     |
|                                                                     | "auth_id - Authorization ID" on page 188                           |
|                                                                     | "client_db_alias - Database Alias Used by Application" on page 190 |
|                                                                     | "codepage_id - ID of Code Page Used by Application" on page 183    |
|                                                                     | "locks_held - Locks Held" on page 297                              |
|                                                                     | "locks_waiting - Current Agents Waiting On Locks" on page 314      |
|                                                                     | "lock_wait_time - Time Waited On Locks" on page 313                |
|                                                                     | "sequence_no - Sequence number monitor element" on page 187        |
|                                                                     | "session_auth_id - Session Authorization ID" on page 189           |
|                                                                     | "status_change_time - Application Status Change Time" on page 183  |
|                                                                     | appl_remote                                                        |
| "create_nickname - Create Nicknames" on page 463                    |                                                                    |
| "create_nickname_time - Create Nickname Response Time" on page 468  |                                                                    |
| "datasource_name - Data Source Name" on page 460                    |                                                                    |
| "db_name - Database Name" on page 172                               |                                                                    |
| "delete_sql_stmts - Deletes" on page 462                            |                                                                    |
| "delete_time - Delete Response Time" on page 467                    |                                                                    |
| "failed_sql_stmts - Failed Statement Operations" on page 364        |                                                                    |
| "insert_sql_stmts - Inserts" on page 461                            |                                                                    |
| "insert_time - Insert Response Time" on page 466                    |                                                                    |
| "passthru_time - Pass-Through Time" on page 468                     |                                                                    |
| "passthru - Pass-Through" on page 463                               |                                                                    |
| "remote_lock_time - Remote Lock Time" on page 469                   |                                                                    |
| "remote_locks - Remote Locks" on page 464                           |                                                                    |
| "rollback_sql_stmts - Rollback Statements Attempted" on page 366    |                                                                    |
| "rows_deleted - Rows Deleted" on page 345                           |                                                                    |
| "rows_inserted - Rows Inserted" on page 345                         |                                                                    |
| "rows_selected - Rows Selected" on page 346                         |                                                                    |
| "rows_updated - Rows Updated" on page 346                           |                                                                    |
| "select_sql_stmts - Select SQL Statements Executed" on page 367     |                                                                    |
| "select_time - Query Response Time" on page 465                     |                                                                    |
| "sp_rows_selected - Rows Returned by Stored Procedures" on page 465 |                                                                    |
| "stored_proc_time - Stored Procedure Time" on page 469              |                                                                    |
| "stored_procs - Stored Procedures" on page 464                      |                                                                    |
| "update_sql_stmts - Updates" on page 462                            |                                                                    |
| "update_time - Update Response Time" on page 466                    |                                                                    |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------|
| bufferpool                   | "block_ios - Number of Block IO Requests" on page 266                                     |
|                              | "bp_name - Buffer Pool Name" on page 263                                                  |
|                              | "bp_id - Buffer pool identifier monitor element" on page 237                              |
|                              | "db_name - Database Name" on page 172                                                     |
|                              | "db_path - Database Path" on page 173                                                     |
|                              | "direct_read_reqs - Direct Read Requests" on page 270                                     |
|                              | "direct_reads - Direct Reads From Database" on page 268                                   |
|                              | "direct_read_time - Direct Read Time" on page 271                                         |
|                              | "direct_write_reqs - Direct Write Requests" on page 270                                   |
|                              | "direct_writes - Direct Writes to Database" on page 269                                   |
|                              | "direct_write_time - Direct Write Time" on page 271                                       |
|                              | "files_closed - Database Files Closed" on page 252                                        |
|                              | "input_db_alias - Input Database Alias" on page 405                                       |
|                              | "pages_from_block_ios - Total Number of Pages Read by Block IO" on page 266               |
|                              | "pages_from_vectored_ios - Total Number of Pages Read by Vectored IO" on page 265         |
|                              | "pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258          |
|                              | "pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252 |
|                              | "pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253               |
|                              | "pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 259   |
|                              | "pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255               |
|                              | "pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254             |
|                              | "pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257                   |
|                              | "pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257                 |
|                              | "pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 259       |
|                              | "pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 255              |
|                              | "pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 256            |
|                              | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                          |
|                              | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                         |
|                              | "pool_data_writes - Buffer Pool Data Writes" on page 240                                  |
|                              | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                        |
|                              | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                       |
|                              | "pool_index_writes - Buffer Pool Index Writes" on page 245                                |
|                              | "pool_read_time - Total Buffer Pool Physical Read Time" on page 250                       |
|                              | "pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 262                       |
|                              | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238           |
|                              | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240          |
|                              | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243         |
|                              | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244        |
|                              | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247        |
|                              | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bufferpool<br>(continued)    | <p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 251</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes" on page 250</p> <p>"vectored_ios - Number of Vectored IO Requests" on page 265</p>                                                                                                                |
| bufferpool_nodeinfo          | <p>"bp_cur_buffsz - Current Size of Buffer Pool" on page 267</p> <p>"bp_new_buffsz - New Buffer Pool Size" on page 267</p> <p>"bp_pages_left_to_remove - Number of Pages Left to Remove" on page 267</p> <p>"bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool" on page 268</p> <p>"node_number - Node Number" on page 198</p>                                                                                                                                            |
| collected                    | <p>"node_number - Node Number" on page 198</p> <p>"server_db2_type - Database Manager Type at Monitored (Server) Node" on page 169</p> <p>"server_instance_name - Server Instance Name" on page 168</p> <p>"server_prdid - Server Product/Version ID" on page 169</p> <p>"server_version - Server Version" on page 170</p> <p>switch_list Monitor switches control data</p> <p>"time_stamp - Snapshot Time" on page 406</p> <p>"time_zone_disp - Time Zone Displacement" on page 172</p> |



Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                  | Monitor element                                                                                  |                                                                 |
|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| db2                                                           | "agents_created_empty_pool - Agents Created Due to Empty Agent Pool" on page 213                 |                                                                 |
|                                                               | "agents_from_pool - Agents Assigned From Pool" on page 212                                       |                                                                 |
|                                                               | "agents_registered - Agents Registered" on page 210                                              |                                                                 |
|                                                               | "agents_registered_top - Maximum Number of Agents Registered" on page 211                        |                                                                 |
|                                                               | "agents_stolen - Stolen Agents" on page 213                                                      |                                                                 |
|                                                               | "agents_waiting_on_token - Agents Waiting for a Token" on page 210                               |                                                                 |
|                                                               | "agents_waiting_top - Maximum Number of Agents Waiting monitor element" on page 211              |                                                                 |
|                                                               | "comm_private_mem - Committed Private Memory" on page 214                                        |                                                                 |
|                                                               | "con_local_databases - Local Databases with Current Connects" on page 208                        |                                                                 |
|                                                               | "coord_agents_top - Maximum Number of Coordinating Agents" on page 213                           |                                                                 |
|                                                               | "db2start_time - Start Database Manager Timestamp" on page 168                                   |                                                                 |
|                                                               | "db_status - Status of Database" on page 175                                                     |                                                                 |
|                                                               | "gw_total_cons - Total Number of Attempted Connections for DB2 Connect" on page 431              |                                                                 |
|                                                               | "gw_cur_cons - Current Number of Connections for DB2 Connect" on page 432                        |                                                                 |
|                                                               | "gw_cons_wait_host - Number of Connections Waiting for the Host to Reply" on page 432            |                                                                 |
|                                                               | "gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request" on page 432 |                                                                 |
|                                                               | "idle_agents - Number of Idle Agents" on page 212                                                |                                                                 |
|                                                               | "last_reset - Last Reset Timestamp" on page 405                                                  |                                                                 |
|                                                               | "local_cons - Local Connections" on page 207                                                     |                                                                 |
|                                                               | "local_cons_in_exec - Local Connections Executing in the Database Manager" on page 208           |                                                                 |
|                                                               | "max_agent_overflows - Maximum Agent Overflows" on page 215                                      |                                                                 |
|                                                               | "num_gw_conn_switches - Connection Switches" on page 215                                         |                                                                 |
|                                                               | "num_nodes_in_db2_instance - Number of Nodes in Partition" on page 406                           |                                                                 |
|                                                               | "piped_sorts_requested - Piped Sorts Requested" on page 221                                      |                                                                 |
|                                                               | "piped_sorts_accepted - Piped Sorts Accepted" on page 222                                        |                                                                 |
|                                                               | "post_threshold_hash_joins - Hash Join Threshold" on page 227                                    |                                                                 |
|                                                               | "post_threshold_olap_funcs - OLAP Function Threshold monitor element" on page 231                |                                                                 |
|                                                               | "post_threshold_sorts - Post Threshold Sorts" on page 220                                        |                                                                 |
|                                                               | "product_name - Product Name" on page 171                                                        |                                                                 |
|                                                               | "rem_cons_in - Remote Connections To Database Manager" on page 206                               |                                                                 |
|                                                               | "rem_cons_in_exec - Remote Connections Executing in the Database Manager" on page 206            |                                                                 |
|                                                               | "service_level - Service Level" on page 170                                                      |                                                                 |
|                                                               | "smallest_log_avail_node - Node with Least Available Log Space" on page 184                      |                                                                 |
|                                                               | "sort_heap_allocated - Total Sort Heap Allocated" on page 220                                    |                                                                 |
|                                                               | "sort_heap_top - Sort private heap high watermark" on page 225                                   |                                                                 |
|                                                               | switch_list Monitor switches control data                                                        |                                                                 |
|                                                               | db_lock_list                                                                                     | "appls_cur_cons - Applications Connected Currently" on page 209 |
|                                                               |                                                                                                  | "db_name - Database Name" on page 172                           |
|                                                               |                                                                                                  | "db_path - Database Path" on page 173                           |
|                                                               |                                                                                                  | "input_db_alias - Input Database Alias" on page 405             |
| "locks_held - Locks Held" on page 297                         |                                                                                                  |                                                                 |
| "locks_waiting - Current Agents Waiting On Locks" on page 314 |                                                                                                  |                                                                 |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                               |
|------------------------------|-----------------------------------------------------------------------------------------------|
| dbase                        | "active_hash_joins - Active hash joins" on page 228                                           |
|                              | "active_olap_funcs - Active OLAP Functions monitor element" on page 232                       |
|                              | "active_sorts - Active Sorts" on page 225                                                     |
|                              | "agents_top - Number of Agents Created" on page 399                                           |
|                              | "appl_id_oldest_xact - Application with Oldest Transaction" on page 184                       |
|                              | "appl_section_inserts - Section Inserts monitor element" on page 285                          |
|                              | "appl_section_lookups - Section Lookups" on page 285                                          |
|                              | "appls_cur_cons - Applications Connected Currently" on page 209                               |
|                              | "appls_in_db2 - Applications Executing in the Database Currently" on page 210                 |
|                              | "async_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 503 |
|                              | "binds_precompiles - Binds/Precompiles Attempted" on page 372                                 |
|                              | "blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element" on page 296      |
|                              | "cat_cache_inserts - Catalog Cache Inserts" on page 273                                       |
|                              | "cat_cache_lookups - Catalog Cache Lookups" on page 272                                       |
|                              | "cat_cache_overflows - Catalog Cache Overflows" on page 274                                   |
|                              | "cat_cache_size_top - Catalog cache high watermark" on page 275                               |
|                              | "catalog_node - Catalog Node Number" on page 176                                              |
|                              | "catalog_node_name - Catalog Node Network Name" on page 175                                   |
|                              | "commit_sql_stmts - Commit Statements Attempted" on page 365                                  |
|                              | "connections_top - Maximum Number of Concurrent Connections" on page 200                      |
|                              | "coord_agents_top - Maximum Number of Coordinating Agents" on page 213                        |
|                              | "db_conn_time - Database Activation Timestamp" on page 174                                    |
|                              | "db_heap_top - Maximum Database Heap Allocated" on page 286                                   |
|                              | "db_location - Database Location" on page 176                                                 |
|                              | "db_name - Database Name" on page 172                                                         |
|                              | "db_path - Database Path" on page 173                                                         |
|                              | "db_status - Status of Database" on page 175                                                  |
|                              | "ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 368                   |
|                              | "deadlocks - Deadlocks Detected" on page 298                                                  |
|                              | "direct_read_reqs - Direct Read Requests" on page 270                                         |
|                              | "direct_read_time - Direct Read Time" on page 271                                             |
|                              | "direct_reads - Direct Reads From Database" on page 268                                       |
|                              | "direct_write_reqs - Direct Write Requests" on page 270                                       |
|                              | "direct_write_time - Direct Write Time" on page 271                                           |
|                              | "direct_writes - Direct Writes to Database" on page 269                                       |
|                              | "dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 364                            |
|                              | "failed_sql_stmts - Failed Statement Operations" on page 364                                  |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                          |
|------------------------------|------------------------------------------------------------------------------------------|
| dbase (continued)            | "files_closed - Database Files Closed" on page 252                                       |
|                              | "hash_join_overflows - Hash Join Overflows" on page 229                                  |
|                              | "hash_join_small_overflows - Hash Join Small Overflows" on page 230                      |
|                              | "input_db_alias - Input Database Alias" on page 405                                      |
|                              | "int_auto_rebinds - Internal Automatic Rebinds" on page 369                              |
|                              | "int_commits - Internal Commits" on page 369                                             |
|                              | "int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 371                |
|                              | "int_rollbacks - Internal Rollbacks" on page 370                                         |
|                              | "int_rows_deleted - Internal Rows Deleted" on page 349                                   |
|                              | "int_rows_inserted - Internal Rows Inserted" on page 351                                 |
|                              | "int_rows_updated - Internal Rows Updated" on page 350                                   |
|                              | "last_backup - Last Backup Timestamp" on page 176                                        |
|                              | "last_reset - Last Reset Timestamp" on page 405                                          |
|                              | "lock_escals - Number of Lock Escalations" on page 299                                   |
|                              | "lock_list_in_use - Total Lock List Memory In Use" on page 297                           |
|                              | "lock_timeouts - Number of Lock Timeouts" on page 305                                    |
|                              | "lock_wait_time - Time Waited On Locks" on page 313                                      |
|                              | "lock_waits - Lock Waits" on page 312                                                    |
|                              | "locks_held - Locks Held" on page 297                                                    |
|                              | "locks_waiting - Current Agents Waiting On Locks" on page 314                            |
|                              | "log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages" on page 290 |
|                              | "log_read_time - Log Read Time" on page 292                                              |
|                              | "log_reads - Number of Log Pages Read" on page 288                                       |
|                              | "log_to_redo_for_recovery - Amount of Log to be Redone for Recovery" on page 291         |
|                              | "log_write_time - Log Write Time" on page 292                                            |
|                              | "log_writes - Number of Log Pages Written" on page 288                                   |
|                              | "num_assoc_agents - Number of Associated Agents" on page 215                             |
|                              | "num_db_storage_paths - Number of automatic storage paths" on page 177                   |
|                              | "num_indoubt_trans - Number of Indoubt Transactions" on page 312                         |
|                              | "num_log_buffer_full - Number of Full Log Buffers" on page 294                           |
|                              | "num_log_data_found_in_buffer - Number of Log Data Found In Buffer" on page 294          |
|                              | "num_log_part_page_io - Number of Partial Log Page Writes" on page 293                   |
|                              | "num_log_read_io - Number of Log Reads" on page 293                                      |
|                              | "num_log_write_io - Number of Log Writes" on page 292                                    |
|                              | "olap_func_overflows - OLAP Function Overflows monitor element" on page 231              |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------|
| dbase (continued)            | "pkg_cache_inserts - Package Cache Inserts" on page 277                                   |
|                              | "pkg_cache_lookups - Package Cache Lookups" on page 276                                   |
|                              | "pkg_cache_num_overflows - Package Cache Overflows" on page 278                           |
|                              | "pkg_cache_size_top - Package cache high watermark" on page 278                           |
|                              | "pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258          |
|                              | "pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252 |
|                              | "pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253               |
|                              | "pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 259   |
|                              | "pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255               |
|                              | "pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254             |
|                              | "pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257                   |
|                              | "pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 259       |
|                              | "pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 255              |
|                              | "pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 256            |
|                              | "pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257                 |
|                              | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                          |
|                              | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                         |
|                              | "pool_data_writes - Buffer Pool Data Writes" on page 240                                  |
|                              | "pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered" on page 261        |
|                              | "pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered" on page 263          |
|                              | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                        |
|                              | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                       |
|                              | "pool_index_writes - Buffer Pool Index Writes" on page 245                                |
|                              | "pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered" on page 260                |
|                              | "pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 262                       |
|                              | "pool_read_time - Total Buffer Pool Physical Read Time" on page 250                       |
|                              | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238           |
|                              | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240          |
|                              | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243         |
|                              | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244        |
|                              | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247        |
|                              | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249       |
|                              | "pool_write_time - Total Buffer Pool Physical Write Time" on page 251                     |
|                              | "pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246                       |
|                              | "pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248                      |
|                              | "pool_xda_writes - Buffer Pool XDA Data Writes" on page 250                               |
|                              | "post_shrthreshold_hash_joins - Post threshold hash joins" on page 228                    |
|                              | "post_shrthreshold_sorts - Post shared threshold sorts" on page 221                       |
|                              | "priv_workspace_num_overflows - Private Workspace Overflows" on page 282                  |
|                              | "priv_workspace_section_inserts - Private Workspace Section Inserts" on page 284          |
|                              | "priv_workspace_section_lookups - Private Workspace Section Lookups" on page 283          |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------|
| dbase (continued)            | "priv_workspace_size_top - Maximum Private Workspace Size" on page 282                                     |
|                              | "rollback_sql_stmts - Rollback Statements Attempted" on page 366                                           |
|                              | "rows_deleted - Rows Deleted" on page 345                                                                  |
|                              | "rows_inserted - Rows Inserted" on page 345                                                                |
|                              | "rows_read - Rows Read" on page 348                                                                        |
|                              | "rows_selected - Rows Selected" on page 346                                                                |
|                              | "rows_updated - Rows Updated" on page 346                                                                  |
|                              | "sec_log_used_top - Maximum Secondary Log Space Used" on page 286                                          |
|                              | "sec_logs_allocated - Secondary Logs Allocated Currently" on page 288                                      |
|                              | "select_sql_stmts - Select SQL Statements Executed" on page 367                                            |
|                              | "server_platform - Server Operating System" on page 171                                                    |
|                              | "shr_workspace_num_overflows - Shared Workspace Overflows" on page 280                                     |
|                              | "shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 281                             |
|                              | "shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 280                             |
|                              | "shr_workspace_size_top - Maximum Shared Workspace Size" on page 279                                       |
|                              | "sort_heap_allocated - Total Sort Heap Allocated" on page 220                                              |
|                              | "sort_overflows - Sort Overflows" on page 224                                                              |
|                              | "sort_shrheap_allocated - Sort Share Heap Currently Allocated" on page 226                                 |
|                              | "sort_shrheap_top - Sort share heap high watermark" on page 226                                            |
|                              | "static_sql_stmts - Static SQL Statements Attempted" on page 363                                           |
|                              | "stats_cache_size - Size of statistics cache monitor element" on page 501                                  |
|                              | "stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 504 |
|                              | "stats_fabrications - Total number of statistics fabrications monitor elements" on page 502                |
|                              | "sync_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 502              |
|                              | "sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 504     |
|                              | "tot_log_used_top - Maximum Total Log Space Used" on page 287                                              |
|                              | "total_cons - Connects Since Database Activation" on page 209                                              |
|                              | "total_hash_joins - Total Hash Joins" on page 227                                                          |
|                              | "total_hash_loops - Total Hash Loops" on page 229                                                          |
|                              | "total_log_available - Total Log Available" on page 290                                                    |
|                              | "total_log_used - Total Log Space Used" on page 289                                                        |
|                              | "total_olap_funcs - Total OLAP Functions monitor element" on page 230                                      |
|                              | "total_sec_cons - Secondary Connections" on page 214                                                       |
|                              | "total_sort_time - Total Sort Time" on page 223                                                            |
|                              | "total_sorts - Total Sorts" on page 222                                                                    |
|                              | "uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 367                                 |
|                              | "unread_prefetch_pages - Unread Prefetch Pages" on page 264                                                |
|                              | "x_lock_escals - Exclusive Lock Escalations" on page 300                                                   |
|                              | "xquery_stmts - XQuery Statements Attempted" on page 373                                                   |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                       | Monitor element                                                     |                                                                    |
|--------------------------------------------------------------------|---------------------------------------------------------------------|--------------------------------------------------------------------|
| dbase_remote                                                       | "commit_sql_stmts - Commit Statements Attempted" on page 365        |                                                                    |
|                                                                    | "create_nickname - Create Nicknames" on page 463                    |                                                                    |
|                                                                    | "create_nickname_time - Create Nickname Response Time" on page 468  |                                                                    |
|                                                                    | "datasource_name - Data Source Name" on page 460                    |                                                                    |
|                                                                    | "db_name - Database Name" on page 172                               |                                                                    |
|                                                                    | "delete_sql_stmts - Deletes" on page 462                            |                                                                    |
|                                                                    | "delete_time - Delete Response Time" on page 467                    |                                                                    |
|                                                                    | "disconnects - Disconnects" on page 461                             |                                                                    |
|                                                                    | "failed_sql_stmts - Failed Statement Operations" on page 364        |                                                                    |
|                                                                    | "insert_sql_stmts - Inserts" on page 461                            |                                                                    |
|                                                                    | "insert_time - Insert Response Time" on page 466                    |                                                                    |
|                                                                    | "passthru_time - Pass-Through Time" on page 468                     |                                                                    |
|                                                                    | "passthru - Pass-Through" on page 463                               |                                                                    |
|                                                                    | "remote_lock_time - Remote Lock Time" on page 469                   |                                                                    |
|                                                                    | "remote_locks - Remote Locks" on page 464                           |                                                                    |
|                                                                    | "rollback_sql_stmts - Rollback Statements Attempted" on page 366    |                                                                    |
|                                                                    | "rows_deleted - Rows Deleted" on page 345                           |                                                                    |
|                                                                    | "rows_inserted - Rows Inserted" on page 345                         |                                                                    |
|                                                                    | "rows_selected - Rows Selected" on page 346                         |                                                                    |
|                                                                    | "rows_updated - Rows Updated" on page 346                           |                                                                    |
|                                                                    | "select_sql_stmts - Select SQL Statements Executed" on page 367     |                                                                    |
|                                                                    | "select_time - Query Response Time" on page 465                     |                                                                    |
|                                                                    | "sp_rows_selected - Rows Returned by Stored Procedures" on page 465 |                                                                    |
|                                                                    | "stored_proc_time - Stored Procedure Time" on page 469              |                                                                    |
|                                                                    | "stored_procs - Stored Procedures" on page 464                      |                                                                    |
|                                                                    | "total_cons - Connects Since Database Activation" on page 209       |                                                                    |
|                                                                    | "update_sql_stmts - Updates" on page 462                            |                                                                    |
|                                                                    | "update_time - Update Response Time" on page 466                    |                                                                    |
|                                                                    | db_storage_group                                                    | "db_storage_path - Automatic storage path" on page 177             |
|                                                                    |                                                                     | "sto_path_free_sz - Automatic Storage Path Free Space" on page 177 |
| "fs_used_size - Amount of Space Used on a File System" on page 178 |                                                                     |                                                                    |
| "fs_total_size - Total Size of a File System" on page 178          |                                                                     |                                                                    |
| "fs_id - Unique File System Identification Number" on page 179     |                                                                     |                                                                    |
| "fs_type - File System Type" on page 179                           |                                                                     |                                                                    |
| "node_number - Node Number" on page 198                            |                                                                     |                                                                    |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dc_s_appl                    | <p>"appl_idle_time - Application Idle Time" on page 204</p> <p>"commit_sql_stmts - Commit Statements Attempted" on page 365</p> <p>"elapsed_exec_time - Statement Execution Elapsed Time" on page 454</p> <p>"failed_sql_stmts - Failed Statement Operations" on page 364</p> <p>"gw_con_time - DB2 Connect Gateway First Connect Initiated" on page 431</p> <p>"gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 433</p> <p>"host_response_time - Host Response Time" on page 455</p> <p>"inbound_bytes_received - Inbound Number of Bytes Received" on page 437</p> <p>"inbound_bytes_sent - Inbound Number of Bytes Sent" on page 439</p> <p>"last_reset - Last Reset Timestamp" on page 405</p> <p>"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 441</p> <p>"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 442</p> <p>"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 443</p> <p>"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 444</p> <p>"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 445</p> <p>"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 445</p> <p>"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 446</p> <p>"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 447</p> <p>"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 448</p> <p>"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 449</p> <p>"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 450</p> |



Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dcs_appl (continued)         | <p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 441</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 442</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 442</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 443</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 444</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 445</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 446</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 447</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 448</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 448</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 449</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 450</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 451</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 451</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 452</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 452</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 452</p> <p>"network_time_top - Maximum Network Time for Statement" on page 453</p> <p>"network_time_bottom - Minimum Network Time for Statement" on page 453</p> <p>"open_cursors - Number of Open Cursors" on page 435</p> <p>"outbound_bytes_received - Outbound Number of Bytes Received" on page 438</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 438</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 201</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 366</p> |



Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                        |
|------------------------------|------------------------------------------------------------------------|
| dcs_appl (continued)         | "rows_selected - Rows Selected" on page 346                            |
|                              | "sql_stmts - Number of SQL Statements Attempted" on page 433           |
|                              | "tpmon_acc_str - TP Monitor Client Accounting String" on page 460      |
|                              | "tpmon_client_app - TP Monitor Client Application Name" on page 459    |
|                              | "tpmon_client_userid - TP Monitor Client User ID" on page 459          |
|                              | "tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 459  |
|                              | "uow_comp_status - Unit of Work Completion Status" on page 203         |
|                              | "uow_elapsed_time - Most Recent Unit of Work Elapsed Time" on page 202 |
|                              | "uow_start_time - Unit of Work Start Timestamp" on page 201            |
|                              | "uow_stop_time - Unit of Work Stop Timestamp" on page 202              |
|                              | "xid - Transaction ID" on page 454                                     |
| dcs_appl_info                | "agent_id - Application Handle (agent ID)" on page 180                 |
|                              | "agent_status - DCS Application Agents" on page 436                    |
|                              | "appl_id - Application ID" on page 185                                 |
|                              | "appl_name - Application Name" on page 185                             |
|                              | "auth_id - Authorization ID" on page 188                               |
|                              | "client_pid - Client Process ID" on page 193                           |
|                              | "client_platform - Client Operating Platform" on page 194              |
|                              | "client_prdid - Client Product/Version ID" on page 189                 |
|                              | "client_protocol - Client Communication Protocol" on page 194          |
|                              | "codepage_id - ID of Code Page Used by Application" on page 183        |
|                              | "dcs_appl_status - DCS Application Status" on page 435                 |
|                              | "dcs_db_name - DCS Database Name" on page 430                          |
|                              | "execution_id - User Login ID" on page 192                             |
|                              | "gw_db_alias - Database Alias at the Gateway" on page 430              |
|                              | "host_ccsid - Host Coded Character Set ID" on page 436                 |
|                              | "host_db_name - Host Database Name" on page 430                        |
|                              | "host_prdid - Host Product/Version ID" on page 190                     |
|                              | "inbound_comm_address - Inbound Communication Address" on page 437     |
|                              | "outbound_appl_id - Outbound Application ID" on page 191               |
|                              | "outbound_comm_address - Outbound Communication Address" on page 437   |
|                              | "outbound_comm_protocol - Outbound Communication Protocol" on page 436 |
|                              | "outbound_sequence_no - Outbound Sequence Number" on page 192          |
|                              | "sequence_no - Sequence number monitor element" on page 187            |
|                              | "status_change_time - Application Status Change Time" on page 183      |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dc_s_dbase                   | <p>"commit_sql_stmts - Commit Statements Attempted" on page 365</p> <p>"con_elapsed_time - Most Recent Connection Elapsed Time" on page 457</p> <p>"con_response_time - Most Recent Response Time for Connect" on page 456</p> <p>"dcs_db_name - DCS Database Name" on page 430</p> <p>"elapsed_exec_time - Statement Execution Elapsed Time" on page 454</p> <p>"failed_sql_stmts - Failed Statement Operations" on page 364</p> <p>"gw_comm_error_time - Communication Error Time" on page 458</p> <p>"gw_comm_errors - Communication Errors" on page 457</p> <p>"gw_con_time - DB2 Connect Gateway First Connect Initiated" on page 431</p> <p>"gw_connections_top - Maximum Number of Concurrent Connections to Host Database" on page 431</p> <p>"gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request" on page 432</p> <p>"gw_cons_wait_host - Number of Connections Waiting for the Host to Reply" on page 432</p> <p>"gw_cur_cons - Current Number of Connections for DB2 Connect" on page 432</p> <p>"gw_total_cons - Total Number of Attempted Connections for DB2 Connect" on page 431</p> <p>"host_db_name - Host Database Name" on page 430</p> <p>"host_response_time - Host Response Time" on page 455</p> <p>"inbound_bytes_received - Inbound Number of Bytes Received" on page 437</p> <p>"last_reset - Last Reset Timestamp" on page 405</p> <p>"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 441</p> <p>"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 442</p> <p>"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 443</p> <p>"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 444</p> <p>"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 445</p> <p>"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 445</p> <p>"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 446</p> <p>"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 447</p> <p>"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 448</p> <p>"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 449</p> <p>"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 450</p> |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dc_s_dbase<br>(continued)    | <p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 441</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 442</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 442</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 443</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 444</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 445</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 446</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 447</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 448</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 448</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 449</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 450</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 451</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 451</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 452</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 452</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 452</p> <p>"network_time_bottom - Minimum Network Time for Statement" on page 453</p> <p>"network_time_top - Maximum Network Time for Statement" on page 453</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 438</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 366</p> <p>"rows_selected - Rows Selected" on page 346</p> <p>"sql_stmts - Number of SQL Statements Attempted" on page 433</p> |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                 | Monitor element                                                                   |
|--------------------------------------------------------------|-----------------------------------------------------------------------------------|
| dcs_stmt                                                     | "blocking_cursor - Blocking Cursor" on page 458                                   |
|                                                              | "creator - Application Creator" on page 378                                       |
|                                                              | "elapsed_exec_time - Statement Execution Elapsed Time" on page 454                |
|                                                              | "fetch_count - Number of Successful Fetches" on page 382                          |
|                                                              | "gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 433 |
|                                                              | "host_response_time - Host Response Time" on page 455                             |
|                                                              | "inbound_bytes_received - Inbound Number of Bytes Received" on page 437           |
|                                                              | "inbound_bytes_sent - Inbound Number of Bytes Sent" on page 439                   |
|                                                              | "num_transmissions_group - Number of Transmissions Group" on page 456             |
|                                                              | "num_transmissions - Number of Transmissions" on page 455                         |
|                                                              | "outbound_bytes_received - Outbound Number of Bytes Received" on page 438         |
|                                                              | "outbound_bytes_sent - Outbound Number of Bytes Sent" on page 438                 |
|                                                              | "package_name - Package Name" on page 375                                         |
|                                                              | "query_card_estimate - Query Number of Rows Estimate" on page 383                 |
|                                                              | "query_cost_estimate - Query Cost Estimate" on page 384                           |
|                                                              | "section_number - Section Number" on page 377                                     |
|                                                              | "stmt_elapsed_time - Most Recent Statement Elapsed Time" on page 380              |
|                                                              | "stmt_operation/operation - Statement Operation" on page 374                      |
|                                                              | "stmt_start - Statement Operation Start Timestamp" on page 378                    |
|                                                              | "stmt_stop - Statement Operation Stop Timestamp" on page 379                      |
| "stmt_text - SQL Statement Text monitor element" on page 380 |                                                                                   |
| detail_log                                                   | "current_active_log - Current Active Log File Number" on page 295                 |
|                                                              | "current_archive_log - Current Archive Log File Number" on page 296               |
|                                                              | "first_active_log - First Active Log File Number" on page 294                     |
|                                                              | "last_active_log - Last Active Log File Number" on page 295                       |
|                                                              | "node_number - Node Number" on page 198                                           |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                            |
|------------------------------|------------------------------------------------------------------------------------------------------------|
| dynsql                       | "fetch_count - Number of Successful Fetches" on page 382                                                   |
|                              | "int_rows_deleted - Internal Rows Deleted" on page 349                                                     |
|                              | "int_rows_inserted - Internal Rows Inserted" on page 351                                                   |
|                              | "int_rows_updated - Internal Rows Updated" on page 350                                                     |
|                              | "num_compilations - Statement Compilations" on page 397                                                    |
|                              | "num_executions - Statement Executions" on page 397                                                        |
|                              | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                                           |
|                              | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                                          |
|                              | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                                         |
|                              | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                                        |
|                              | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238                            |
|                              | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240                           |
|                              | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243                          |
|                              | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244                         |
|                              | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247                         |
|                              | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249                        |
|                              | "pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246                                        |
|                              | "pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248                                       |
|                              | "prep_time_best - Statement best preparation time monitor element" on page 398                             |
|                              | "prep_time_worst - Statement worst preparation time monitor element" on page 398                           |
|                              | "rows_read - Rows Read" on page 348                                                                        |
|                              | "rows_written - Rows Written" on page 347                                                                  |
|                              | "sort_overflows - Sort Overflows" on page 224                                                              |
|                              | "stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 504 |
|                              | "stmt_pkgcache_id - Statement package cache identifier" on page 389                                        |
|                              | "stmt_sorts - Statement Sorts" on page 381                                                                 |
|                              | "stmt_text - SQL Statement Text monitor element" on page 380                                               |
|                              | "sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 504     |
|                              | "total_exec_time - Elapsed Statement Execution Time" on page 398                                           |
|                              | "total_sort_time - Total Sort Time" on page 223                                                            |
|                              | "total_sys_cpu_time - Total System CPU for a Statement" on page 404                                        |
|                              | "total_usr_cpu_time - Total User CPU for a Statement" on page 404                                          |
|                              | "insert_timestamp - Statement insert timestamp monitor element" on page 380                                |
| dynsql_list                  | "db_name - Database Name" on page 172                                                                      |
|                              | "db_path - Database Path" on page 173                                                                      |
| fcm                          | "buff_free - FCM Buffers Currently Free" on page 232                                                       |
|                              | "buff_free_bottom - Minimum FCM Buffers Free" on page 233                                                  |
|                              | "ch_free - Channels Currently Free" on page 234                                                            |
|                              | "ch_free_bottom - Minimum Channels Free" on page 235                                                       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                            | Monitor element                                                             |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| fcm_node                                                                | "connection_status - Connection Status" on page 233                         |
|                                                                         | "node_number - Node Number" on page 198                                     |
|                                                                         | "total_buffers_sent - Total FCM Buffers Sent" on page 234                   |
|                                                                         | "total_buffers_rcvd - Total FCM Buffers Received" on page 234               |
| hadr                                                                    | "hadr_connect_status - HADR Connection Status monitor element" on page 420  |
|                                                                         | "hadr_connect_time - HADR Connection Time monitor element" on page 421      |
|                                                                         | "hadr_heartbeat - HADR Heartbeat monitor element" on page 422               |
|                                                                         | "hadr_local_host - HADR Local Host monitor element" on page 422             |
|                                                                         | "hadr_local_service - HADR Local Service monitor element" on page 423       |
|                                                                         | "hadr_log_gap - HADR Log Gap" on page 428                                   |
|                                                                         | "hadr_primary_log_file - HADR Primary Log File monitor element" on page 425 |
|                                                                         | "hadr_primary_log_lsn - HADR Primary Log LSN monitor element" on page 426   |
|                                                                         | "hadr_primary_log_page - HADR Primary Log Page monitor element" on page 426 |
|                                                                         | "hadr_remote_host - HADR Remote Host monitor element" on page 423           |
|                                                                         | "hadr_remote_instance - HADR Remote Instance monitor element" on page 424   |
|                                                                         | "hadr_remote_service - HADR Remote Service monitor element" on page 424     |
|                                                                         | "hadr_role - HADR Role" on page 418                                         |
|                                                                         | "hadr_standby_log_file - HADR Standby Log File monitor element" on page 427 |
|                                                                         | "hadr_standby_log_lsn - HADR Standby Log LSN monitor element" on page 428   |
|                                                                         | "hadr_standby_log_page - HADR Standby Log Page monitor element" on page 427 |
| "hadr_state - HADR State monitor element" on page 419                   |                                                                             |
| "hadr_syncmode - HADR Synchronization Mode monitor element" on page 420 |                                                                             |
| "hadr_timeout - HADR Timeout monitor element" on page 425               |                                                                             |
| lock                                                                    | "data_partition_id - Data partition identifier monitor element" on page 298 |
|                                                                         | "lock_attributes - Lock Attributes" on page 309                             |
|                                                                         | "lock_count - Lock Count" on page 311                                       |
|                                                                         | "lock_current_mode - Original Lock Mode Before Conversion" on page 312      |
|                                                                         | "lock_escalation - Lock Escalation" on page 306                             |
|                                                                         | "lock_hold_count - Lock Hold Count" on page 311                             |
|                                                                         | "lock_mode - Lock Mode" on page 301                                         |
|                                                                         | "lock_name - Lock Name" on page 309                                         |
|                                                                         | "lock_object_name - Lock Object Name" on page 304                           |
|                                                                         | "lock_object_type - Lock Object Type Waited On" on page 303                 |
|                                                                         | "lock_release_flags - Lock Release Flags" on page 310                       |
|                                                                         | "lock_status - Lock Status" on page 302                                     |
|                                                                         | "node_number - Node Number" on page 198                                     |
|                                                                         | "table_file_id - Table File ID" on page 351                                 |
|                                                                         | "table_name - Table Name" on page 343                                       |
| "table_schema - Table Schema Name" on page 344                          |                                                                             |
| "tablespace_name - Table Space Name" on page 320                        |                                                                             |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                                | Monitor element                                                                 |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| lock_wait                                                                   | "agent_id_holding_lock - Agent ID Holding Lock" on page 316                     |
|                                                                             | "appl_id_holding_lk - Application ID Holding Lock" on page 316                  |
|                                                                             | "lock_attributes - Lock Attributes" on page 309                                 |
|                                                                             | "lock_current_mode - Original Lock Mode Before Conversion" on page 312          |
|                                                                             | "lock_escalation - Lock Escalation" on page 306                                 |
|                                                                             | "lock_mode - Lock Mode" on page 301                                             |
|                                                                             | "lock_mode_requested - Lock Mode Requested" on page 306                         |
|                                                                             | "lock_name - Lock Name" on page 309                                             |
|                                                                             | "lock_object_type - Lock Object Type Waited On" on page 303                     |
|                                                                             | "lock_release_flags - Lock Release Flags" on page 310                           |
|                                                                             | "lock_wait_start_time - Lock Wait Start Timestamp" on page 315                  |
|                                                                             | "node_number - Node Number" on page 198                                         |
|                                                                             | "ss_number - Subsection Number" on page 392                                     |
|                                                                             | "table_name - Table Name" on page 343                                           |
| "table_schema - Table Schema Name" on page 344                              |                                                                                 |
| "tablespace_name - Table Space Name" on page 320                            |                                                                                 |
| "data_partition_id - Data partition identifier monitor element" on page 298 |                                                                                 |
| memory_pool                                                                 | "node_number - Node Number" on page 198                                         |
|                                                                             | "pool_cur_size - Current Size of Memory Pool" on page 218                       |
|                                                                             | "pool_id - Memory Pool Identifier" on page 216                                  |
|                                                                             | "pool_secondary_id - Memory Pool Secondary Identifier" on page 217              |
|                                                                             | "pool_config_size - Configured Size of Memory Pool" on page 218                 |
| "pool_watermark - Memory Pool Watermark" on page 219                        |                                                                                 |
| progress                                                                    | "progress_completed_units - Completed Progress Work Units" on page 417          |
|                                                                             | "progress_description - Progress Description" on page 415                       |
|                                                                             | "progress_seq_num - Progress Sequence Number" on page 415                       |
|                                                                             | "progress_start_time - Progress Start Time" on page 416                         |
|                                                                             | "progress_total_units - Total Progress Work Units" on page 416                  |
| "progress_work_metric - Progress Work Metric" on page 416                   |                                                                                 |
| progress_list                                                               | "progress_list_cur_seq_num - Current Progress List Sequence Number" on page 415 |
|                                                                             | "progress_list_attr - Current Progress List Attributes" on page 417             |
| rollforward                                                                 | "node_number - Node Number" on page 198                                         |
|                                                                             | "rf_type - Rollforward Type" on page 319                                        |
|                                                                             | "rf_log_num - Log Being Rolled Forward" on page 319                             |
|                                                                             | "rf_status - Log Phase" on page 320                                             |
|                                                                             | "rf_timestamp - Rollforward Timestamp" on page 318                              |
| "ts_name - Tablespace Being Rolled Forward" on page 319                     |                                                                                 |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                     |
|------------------------------|-------------------------------------------------------------------------------------|
| stmt                         | "agents_top - Number of Agents Created" on page 399                                 |
|                              | "blocking_cursor - Blocking Cursor" on page 458                                     |
|                              | "consistency_token - Package Consistency Token" on page 376                         |
|                              | "creator - Application Creator" on page 378                                         |
|                              | "cursor_name - Cursor Name" on page 377                                             |
|                              | "degree_parallelism - Degree of Parallelism" on page 399                            |
|                              | "fetch_count - Number of Successful Fetches" on page 382                            |
|                              | "int_rows_deleted - Internal Rows Deleted" on page 349                              |
|                              | "int_rows_inserted - Internal Rows Inserted" on page 351                            |
|                              | "int_rows_updated - Internal Rows Updated" on page 350                              |
|                              | "num_agents - Number of Agents Working on a Statement" on page 399                  |
|                              | "package_name - Package Name" on page 375                                           |
|                              | "package_version_id - Package Version" on page 376                                  |
|                              | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                    |
|                              | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                   |
|                              | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                  |
|                              | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                 |
|                              | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238     |
|                              | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240    |
|                              | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243   |
|                              | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244  |
|                              | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247  |
|                              | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249 |
|                              | "pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246                 |
|                              | "pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248                |
|                              | "query_card_estimate - Query Number of Rows Estimate" on page 383                   |
|                              | "query_cost_estimate - Query Cost Estimate" on page 384                             |
|                              | "rows_read - Rows Read" on page 348                                                 |
|                              | "rows_written - Rows Written" on page 347                                           |
|                              | "section_number - Section Number" on page 377                                       |
|                              | "sort_overflows - Sort Overflows" on page 224                                       |
|                              | "stmt_elapsed_time - Most Recent Statement Elapsed Time" on page 380                |
|                              | "stmt_node_number - Statement Node" on page 372                                     |
|                              | "stmt_operation/operation - Statement Operation" on page 374                        |
|                              | "stmt_sorts - Statement Sorts" on page 381                                          |
|                              | "stmt_start - Statement Operation Start Timestamp" on page 378                      |
|                              | "stmt_stop - Statement Operation Stop Timestamp" on page 379                        |
|                              | "stmt_sys_cpu_time - System CPU Time used by Statement" on page 401                 |
|                              | "stmt_text - SQL Statement Text monitor element" on page 380                        |
|                              | "stmt_type - Statement Type" on page 373                                            |
|                              | "stmt_usr_cpu_time - User CPU Time used by Statement" on page 401                   |
|                              | "total_sort_time - Total Sort Time" on page 223                                     |



Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                         |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| stmt_transmissions           | "elapsed_exec_time - Statement Execution Elapsed Time" on page 454                                                                      |
|                              | "host_response_time - Host Response Time" on page 455                                                                                   |
|                              | "max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 441                         |
|                              | "max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 442                       |
|                              | "max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 443                       |
|                              | "max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 444                     |
|                              | "max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 445                    |
|                              | "max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 445                    |
|                              | "max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 446                    |
|                              | "max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 447                  |
|                              | "max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 448 |
|                              | "max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 449 |
|                              | "max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 450                    |
|                              | "max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 441                                 |
|                              | "max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 442                               |
|                              | "max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 442                               |
|                              | "max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 443                             |
|                              | "max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 444                            |
|                              | "max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 445                            |
|                              | "max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 446                            |
|                              | "max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 447                          |
|                              | "max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 448                         |
|                              | "max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 448                         |
|                              | "max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 449                            |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                                 | Monitor element                                                                                       |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| stmt_transmissions<br>(continued)                                            | "max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 450            |
|                                                                              | "max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 451       |
|                                                                              | "max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 451     |
|                                                                              | "max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 452  |
|                                                                              | "max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 452 |
|                                                                              | "max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 452  |
|                                                                              | "network_time_top - Maximum Network Time for Statement" on page 453                                   |
|                                                                              | "network_time_bottom - Minimum Network Time for Statement" on page 453                                |
|                                                                              | "outbound_bytes_received - Outbound Number of Bytes Received" on page 438                             |
|                                                                              | "outbound_bytes_sent - Outbound Number of Bytes Sent" on page 438                                     |
|                                                                              | "outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent" on page 439                         |
|                                                                              | "outbound_bytes_received_top - Maximum Outbound Number of Bytes Received" on page 440                 |
|                                                                              | "outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent" on page 440                      |
|                                                                              | "outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received" on page 440              |
|                                                                              | "sql_chains - Number of SQL Chains Attempted" on page 434                                             |
| "sql_stmts - Number of SQL Statements Attempted" on page 433                 |                                                                                                       |
| subsection                                                                   | "rows_read - Rows Read" on page 348                                                                   |
|                                                                              | "rows_written - Rows Written" on page 347                                                             |
|                                                                              | "ss_exec_time - Subsection Execution Elapsed Time" on page 393                                        |
|                                                                              | "ss_node_number - Subsection Node Number" on page 392                                                 |
|                                                                              | "ss_number - Subsection Number" on page 392                                                           |
|                                                                              | "ss_status - Subsection Status" on page 393                                                           |
|                                                                              | "ss_sys_cpu_time - System CPU Time used by Subsection" on page 403                                    |
|                                                                              | "ss_usr_cpu_time - User CPU Time used by Subsection" on page 403                                      |
|                                                                              | "tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed" on page 395                    |
|                                                                              | "tq_id_waiting_on - Waited on Node on a Tablequeue" on page 397                                       |
|                                                                              | "tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows" on page 396                     |
|                                                                              | "tq_node_waited_for - Waited for Node on a Tablequeue" on page 394                                    |
|                                                                              | "tq_rows_read - Number of Rows Read from Tablequeues" on page 395                                     |
|                                                                              | "tq_rows_written - Number of Rows Written to Tablequeues" on page 396                                 |
|                                                                              | "tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed" on page 394                      |
| "tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue" on page 394 |                                                                                                       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                                      | Monitor element                                                                        |
|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| table                                                                             | "data_object_pages - Data Object Pages" on page 352                                    |
|                                                                                   | "data_partition_id - Data partition identifier monitor element" on page 298            |
|                                                                                   | "index_object_pages - Index Object Pages" on page 353                                  |
|                                                                                   | "lob_object_pages - LOB Object Pages" on page 353                                      |
|                                                                                   | "long_object_pages - Long Object Pages" on page 354                                    |
|                                                                                   | "overflow_accesses - Accesses to Overflowed Records" on page 349                       |
|                                                                                   | "page_reorgs - Page Reorganizations" on page 352                                       |
|                                                                                   | "rows_read - Rows Read" on page 348                                                    |
|                                                                                   | "rows_written - Rows Written" on page 347                                              |
|                                                                                   | "table_file_id - Table File ID" on page 351                                            |
|                                                                                   | "table_name - Table Name" on page 343                                                  |
|                                                                                   | "table_schema - Table Schema Name" on page 344                                         |
|                                                                                   | "tablespace_id - Table Space Identification" on page 320                               |
|                                                                                   | "data_partition_id - Data partition identifier monitor element" on page 298            |
|                                                                                   | "table_type - Table Type" on page 342                                                  |
| "xda_object_pages - XDA Object Pages" on page 354                                 |                                                                                        |
| table_list                                                                        | "db_conn_time - Database Activation Timestamp" on page 174                             |
|                                                                                   | "db_name - Database Name" on page 172                                                  |
|                                                                                   | "db_path - Database Path" on page 173                                                  |
|                                                                                   | "input_db_alias - Input Database Alias" on page 405                                    |
|                                                                                   | "last_reset - Last Reset Timestamp" on page 405                                        |
| table_reorg                                                                       | "data_partition_id - Data partition identifier monitor element" on page 298            |
|                                                                                   | "reorg_completion - Reorganization Completion Flag" on page 358                        |
|                                                                                   | "reorg_current_counter - Reorganize Progress" on page 357                              |
|                                                                                   | "reorg_end - Table Reorganize End Time" on page 358                                    |
|                                                                                   | "reorg_index_id - Index Used to Reorganize the Table" on page 359                      |
|                                                                                   | "reorg_max_counter - Total Amount of Reorganization" on page 357                       |
|                                                                                   | "reorg_max_phase - Maximum Reorganize Phase" on page 357                               |
|                                                                                   | "reorg_phase - Reorganize Phase" on page 356                                           |
|                                                                                   | "reorg_phase_start - Reorganize Phase Start Time" on page 356                          |
|                                                                                   | "reorg_start - Table Reorganize Start Time" on page 358                                |
|                                                                                   | "reorg_status - Table Reorganize Status" on page 355                                   |
|                                                                                   | "reorg_tbsp_id - Table Space Where Table or Data partition is Reorganized" on page 359 |
|                                                                                   | "reorg_type - Table Reorganize Attributes" on page 355                                 |
|                                                                                   | "reorg_rows_compressed - Rows Compressed" on page 359                                  |
| "reorg_rows_rejected_for_compression - Rows Rejected for Compression" on page 360 |                                                                                        |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------|
| tablespace                   | "direct_read_reqs - Direct Read Requests" on page 270                                     |
|                              | "direct_read_time - Direct Read Time" on page 271                                         |
|                              | "direct_reads - Direct Reads From Database" on page 268                                   |
|                              | "direct_write_reqs - Direct Write Requests" on page 270                                   |
|                              | "direct_write_time - Direct Write Time" on page 271                                       |
|                              | "direct_writes - Direct Writes to Database" on page 269                                   |
|                              | "files_closed - Database Files Closed" on page 252                                        |
|                              | "fs_caching - File System Caching" on page 331                                            |
|                              | "pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258          |
|                              | "pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252 |
|                              | "pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253               |
|                              | "pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 259   |
|                              | "pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255               |
|                              | "pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254             |
|                              | "pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257                   |
|                              | "pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257                 |
|                              | "pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 259       |
|                              | "pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 255              |
|                              | "pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 256            |
|                              | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                          |
|                              | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                         |
|                              | "pool_data_writes - Buffer Pool Data Writes" on page 240                                  |
|                              | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                        |
|                              | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                       |
|                              | "pool_index_writes - Buffer Pool Index Writes" on page 245                                |
|                              | "pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 262                       |
|                              | "pool_read_time - Total Buffer Pool Physical Read Time" on page 250                       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                          | Monitor element                                                                       |
|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| tablespace<br>(continued)                                             | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238       |
|                                                                       | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240      |
|                                                                       | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243     |
|                                                                       | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244    |
|                                                                       | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247    |
|                                                                       | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249   |
|                                                                       | "pool_write_time - Total Buffer Pool Physical Write Time" on page 251                 |
|                                                                       | "pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246                   |
|                                                                       | "pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248                  |
|                                                                       | "pool_xda_writes - Buffer Pool XDA Data Writes" on page 250                           |
|                                                                       | "tablespace_auto_resize_enabled - Auto-resize enabled" on page 332                    |
|                                                                       | "tablespace_content_type - Table Space Contents Type" on page 321                     |
|                                                                       | "tablespace_cur_pool_id - Buffer Pool Currently Being Used" on page 324               |
|                                                                       | "tablespace_extent_size - Table Space Extent Size" on page 323                        |
|                                                                       | "tablespace_id - Table Space Identification" on page 320                              |
|                                                                       | "tablespace_name - Table Space Name" on page 320                                      |
|                                                                       | "tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup" on page 324 |
|                                                                       | "tablespace_page_size - Table Space Page Size" on page 323                            |
|                                                                       | "tablespace_prefetch_size - Table Space Prefetch Size" on page 323                    |
|                                                                       | "tablespace_rebalancer_mode - Rebalancer Mode" on page 327                            |
| "tablespace_type - Table Space Type" on page 321                      |                                                                                       |
| "tablespace_using_auto_storage - Using automatic storage" on page 331 |                                                                                       |
| tablespace_container                                                  | "container_accessible - Accessibility of Container" on page 339                       |
|                                                                       | "container_id - Container Identification" on page 336                                 |
|                                                                       | "container_name - Container Name" on page 337                                         |
|                                                                       | "container_stripe_set - Stripe Set" on page 338                                       |
|                                                                       | "container_total_pages - Total Pages in Container" on page 338                        |
|                                                                       | "container_type - Container Type" on page 337                                         |
| "container_usable_pages - Usable Pages in Container" on page 338      |                                                                                       |
| tablespace_list                                                       | "db_conn_time - Database Activation Timestamp" on page 174                            |
|                                                                       | "db_name - Database Name" on page 172                                                 |
|                                                                       | "db_path - Database Path" on page 173                                                 |
|                                                                       | "input_db_alias - Input Database Alias" on page 405                                   |
|                                                                       | "last_reset - Last Reset Timestamp" on page 405                                       |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups                                    | Monitor element                                                                                                   |
|-----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| tablespace_nodeinfo                                             | "tablespace_current_size - Current table space size" on page 332                                                  |
|                                                                 | "tablespace_free_pages - Free Pages in Table Space" on page 326                                                   |
|                                                                 | "tablespace_increase_size - Increase size in bytes" on page 333                                                   |
|                                                                 | "tablespace_increase_size_percent - Increase size by percent" on page 334                                         |
|                                                                 | "tablespace_initial_size - Initial table space size" on page 332                                                  |
|                                                                 | "tablespace_last_resize_failed - Last resize attempt failed" on page 334                                          |
|                                                                 | "tablespace_last_resize_time - Time of last successful resize" on page 334                                        |
|                                                                 | "tablespace_max_size - Maximum table space size" on page 333                                                      |
|                                                                 | "tablespace_min_recovery_time - Minimum Recovery Time For Rollforward" on page 330                                |
|                                                                 | "tablespace_num_containers - Number of Containers in Table Space" on page 331                                     |
|                                                                 | "tablespace_num_quiescers - Number of Quiescers" on page 329                                                      |
|                                                                 | "tablespace_num_ranges - Number of Ranges in the Table Space Map" on page 331                                     |
|                                                                 | "tablespace_page_top - Table space high watermark" on page 326                                                    |
|                                                                 | "tablespace_pending_free_pages - Pending Free Pages in Table Space" on page 326                                   |
|                                                                 | "tablespace_prefetch_size - Table Space Prefetch Size" on page 323                                                |
|                                                                 | "tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed" on page 328            |
|                                                                 | "tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer" on page 328 |
|                                                                 | "tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer" on page 328                       |
|                                                                 | "tablespace_rebalancer_priority - Current Rebalancer Priority" on page 329                                        |
|                                                                 | "tablespace_rebalancer_restart_time - Rebalancer Restart Time" on page 327                                        |
|                                                                 | "tablespace_rebalancer_start_time - Rebalancer Start Time" on page 327                                            |
|                                                                 | "tablespace_state - Table Space State" on page 322                                                                |
|                                                                 | "tablespace_state_change_object_id - State Change Object Identification" on page 330                              |
|                                                                 | "tablespace_state_change_ts_id - State Change Table Space Identification" on page 330                             |
|                                                                 | "tablespace_total_pages - Total Pages in Table Space" on page 324                                                 |
|                                                                 | "tablespace_usable_pages - Usable Pages in Table Space" on page 325                                               |
| "tablespace_used_pages - Used Pages in Table Space" on page 325 |                                                                                                                   |
| tablespace_quiescer                                             | "quiescer_agent_id - Quiescer Agent Identification" on page 335                                                   |
|                                                                 | "quiescer_auth_id - Quiescer User Authorization Identification" on page 335                                       |
|                                                                 | "quiescer_obj_id - Quiescer Object Identification" on page 336                                                    |
|                                                                 | "quiescer_state - Quiescer State" on page 336                                                                     |
|                                                                 | "quiescer_ts_id - Quiescer Table Space Identification" on page 335                                                |

Table 15. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

| Snapshot logical data groups | Monitor element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tablespace_range             | <p>“range_adjustment - Range Adjustment” on page 341</p> <p>“range_container_id - Range Container” on page 342</p> <p>“range_end_stripe - End Stripe” on page 341</p> <p>“range_max_extent - Maximum Extent in Range” on page 340</p> <p>“range_max_page_number - Maximum Page in Range” on page 340</p> <p>“range_num_containers - Number of Containers in Range” on page 341</p> <p>“range_number - Range Number” on page 340</p> <p>“range_offset - Range Offset” on page 342</p> <p>“range_start_stripe - Start Stripe” on page 341</p> <p>“range_stripe_set_number - Stripe Set Number” on page 339</p> |
| utility_info                 | <p>“utility_dbname - Database Operated on by Utility” on page 412</p> <p>“utility_id - Utility ID” on page 412</p> <p>“utility_invoker_type - Utility Invoker Type” on page 414</p> <p>“utility_state - Utility State” on page 414</p> <p>“utility_type - Utility Type” on page 413</p> <p>“utility_priority - Utility Priority” on page 413</p> <p>“utility_start_time - Utility Start Time” on page 413</p> <p>“utility_description - Utility Description” on page 413</p> <p>“node_number - Node Number” on page 198</p>                                                                                  |

## Event type mappings to logical data groups

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 meta-data.

Table 16. 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 17. 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 18. 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 | event_xact         | Transaction level data.                                                                                   |
| Connection Event  | event_conn         | Connection level data.                                                                                    |
| Deadlock Event    | event_deadlock     | Deadlock level data.                                                                                      |



Table 18. Event Monitor Data Stream: Contents Section (continued)

| Event type                               | Logical data group          | Available information                                                                                                                                                                     |
|------------------------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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           | event_stmt_history          | List of statements executed in any unit of work that was involved in a deadlock.                                                                                                          |
| Deadlocks with details history values    | 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_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_threshold_violations  | Information identifying the threshold violated and the time of violation.                                                                                                                 |

## Epilog

The Epilog information is generated during database deactivation (last application finished disconnecting):

Table 19. 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 19. Event Monitor Data Stream: Epilog Section (continued)

| Event type           | Logical data group | Available information   |
|----------------------|--------------------|-------------------------|
| Table Space<br>Event | event_tablespace   | Table space level data. |
| Table Event          | event_table        | Table level data.       |

---

## Event monitor logical data groups and monitor elements

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

Table 20. Event Monitor Logical Data Groups and Monitor Elements

| Event logical data groups | Monitor element name                                                                       |
|---------------------------|--------------------------------------------------------------------------------------------|
| event_activity            | "activity_type - Activity type monitor element" on page 472                                |
|                           | "activate_timestamp - Activate timestamp monitor element" on page 470                      |
|                           | "activity_secondary_id - Activity secondary ID monitor element" on page 471                |
|                           | "coord_partition_num - Coordinator partition number monitor element" on page 478           |
|                           | "partial_record - Partial Record monitor element" on page 408                              |
|                           | "workload_id - Workload ID monitor element" on page 500                                    |
|                           | "workload_occurrence_id - Workload occurrence identifier monitor element" on page 501      |
|                           | "service_superclass_name - Service superclass name monitor element" on page 491            |
|                           | "service_subclass_name - Service subclass name monitor element" on page 491                |
|                           | "db_work_action_set_id - Database work action set ID monitor element" on page 482          |
|                           | "db_work_class_id - Database work class ID monitor element" on page 483                    |
|                           | "sc_work_action_set_id - Service class work action set ID monitor element" on page 489     |
|                           | "sc_work_class_id - Service class work class ID monitor element" on page 490               |
|                           | "agent_id - Application Handle (agent ID)" on page 180                                     |
|                           | "appl_id - Application ID" on page 185                                                     |
|                           | "appl_name - Application Name" on page 185                                                 |
|                           | "uow_id - Unit of work ID monitor element" on page 497                                     |
|                           | "activity_id - Activity ID monitor element" on page 470                                    |
|                           | "parent_uow_id - Parent unit of work ID monitor element" on page 486                       |
|                           | "parent_activity_id - Parent activity ID monitor element" on page 485                      |
|                           | "session_auth_id - Session Authorization ID" on page 189                                   |
|                           | "time_created - Time created monitor element" on page 496                                  |
|                           | "time_started - Time started monitor element" on page 497                                  |
|                           | "time_completed - Time completed monitor element" on page 496                              |
|                           | "act_exec_time - Activity execution time monitor element" on page 472                      |
|                           | "tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 459                      |
|                           | "tpmon_client_app - TP Monitor Client Application Name" on page 459                        |
|                           | "tpmon_client_userid - TP Monitor Client User ID" on page 459                              |
|                           | "tpmon_acc_str - TP Monitor Client Accounting String" on page 460                          |
|                           | "arm_correlator - Application response measurement correlator monitor element" on page 473 |
|                           | "sqlca - SQL Communications Area (SQLCA)" on page 383                                      |
|                           | "total_sorts - Total Sorts" on page 222                                                    |
|                           | "total_sort_time - Total Sort Time" on page 223                                            |
|                           | "sort_overflows - Sort Overflows" on page 224                                              |
|                           | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                           |
|                           | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                          |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                       | Monitor element name                                                                |
|-----------------------------------------------------------------|-------------------------------------------------------------------------------------|
| event_activity (continued)                                      | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                  |
|                                                                 | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                 |
|                                                                 | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238     |
|                                                                 | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240    |
|                                                                 | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243   |
|                                                                 | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244  |
|                                                                 | "pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 246                 |
|                                                                 | "pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 248                |
|                                                                 | "pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 247  |
|                                                                 | "pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 249 |
|                                                                 | "query_cost_estimate - Query Cost Estimate" on page 384                             |
|                                                                 | "query_card_estimate - Query Number of Rows Estimate" on page 383                   |
|                                                                 | "rows_returned - Rows returned monitor element" on page 488                         |
|                                                                 | "rows_fetched - Rows fetched monitor element" on page 487                           |
|                                                                 | "rows_modified - Rows modified monitor element" on page 488                         |
|                                                                 | "system_cpu_time - System CPU Time" on page 402                                     |
|                                                                 | "user_cpu_time - User CPU Time" on page 402                                         |
|                                                                 | "prep_time - Preparation time monitor element" on page 486                          |
| event_activitystmt                                              | "activate_timestamp - Activate timestamp monitor element" on page 470               |
|                                                                 | "appl_id - Application ID" on page 185                                              |
|                                                                 | "uow_id - Unit of work ID monitor element" on page 497                              |
|                                                                 | "activity_id - Activity ID monitor element" on page 470                             |
|                                                                 | "activity_secondary_id - Activity secondary ID monitor element" on page 471         |
|                                                                 | "stmt_first_use_time - Statement first use time" on page 385                        |
|                                                                 | "stmt_last_use_time - Statement last use time monitor element" on page 385          |
|                                                                 | "stmt_lock_timeout - Statement lock timeout" on page 386                            |
|                                                                 | "stmt_isolation - Statement isolation" on page 386                                  |
|                                                                 | "package_name - Package Name" on page 375                                           |
|                                                                 | "package_version_id - Package Version" on page 376                                  |
|                                                                 | "section_number - Section Number" on page 377                                       |
|                                                                 | "creator - Application Creator" on page 378                                         |
|                                                                 | "stmt_type - Statement Type" on page 373                                            |
|                                                                 | "stmt_text - SQL Statement Text monitor element" on page 380                        |
|                                                                 | "comp_env_desc - Compilation environment handle" on page 389                        |
|                                                                 | "stmt_nest_level - Statement nesting level" on page 387                             |
|                                                                 | "stmt_query_id - Statement query identifier" on page 388                            |
|                                                                 | "stmt_source_id - Statement source identifier" on page 388                          |
|                                                                 | "stmt_pkgcache_id - Statement package cache identifier" on page 389                 |
| "section_env - Section environment monitor element" on page 490 |                                                                                     |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                      |
|---------------------------|-------------------------------------------------------------------------------------------|
| event_activityvals        | "activate_timestamp - Activate timestamp monitor element" on page 470                     |
|                           | "stmt_value_type - Value type" on page 390                                                |
|                           | "appl_id - Application ID" on page 185                                                    |
|                           | "uow_id - Unit of work ID monitor element" on page 497                                    |
|                           | "activity_id - Activity ID monitor element" on page 470                                   |
|                           | "activity_secondary_id - Activity secondary ID monitor element" on page 471               |
|                           | "stmt_value_isnull - Value has null value" on page 390                                    |
|                           | "stmt_value_data - Value data" on page 390                                                |
|                           | "stmt_value_index - Value index" on page 391                                              |
|                           | "stmt_value_isreopt - Variable used for statement reoptimization" on page 391             |
| event_bufferpool          | "bp_name - Buffer Pool Name" on page 263                                                  |
|                           | "bp_id - Buffer pool identifier monitor element" on page 237                              |
|                           | "db_name - Database Name" on page 172                                                     |
|                           | "db_path - Database Path" on page 173                                                     |
|                           | "direct_read_reqs - Direct Read Requests" on page 270                                     |
|                           | "direct_read_time - Direct Read Time" on page 271                                         |
|                           | "direct_reads - Direct Reads From Database" on page 268                                   |
|                           | "direct_write_reqs - Direct Write Requests" on page 270                                   |
|                           | "direct_write_time - Direct Write Time" on page 271                                       |
|                           | "direct_writes - Direct Writes to Database" on page 269                                   |
|                           | "event_time - Event Time" on page 409                                                     |
|                           | "evmon_activates - Number of Event Monitor Activations" on page 410                       |
|                           | "evmon_flushes - Number of Event Monitor Flushes" on page 410                             |
|                           | "files_closed - Database Files Closed" on page 252                                        |
|                           | "partial_record - Partial Record monitor element" on page 408                             |
|                           | "pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258          |
|                           | "pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252 |
|                           | "pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253               |
|                           | "pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255               |
|                           | "pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254             |
|                           | "pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257                   |
|                           | "pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257                 |
|                           | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                          |
|                           | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                         |
|                           | "pool_data_writes - Buffer Pool Data Writes" on page 240                                  |
|                           | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                        |
|                           | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                       |
|                           | "pool_index_writes - Buffer Pool Index Writes" on page 245                                |
|                           | "pool_read_time - Total Buffer Pool Physical Read Time" on page 250                       |
|                           | "pool_write_time - Total Buffer Pool Physical Write Time" on page 251                     |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                        |
|---------------------------|-----------------------------------------------------------------------------|
| event_conn                | "acc_curs_blk - Accepted Block Cursor Requests" on page 362                 |
|                           | "agent_id - Application Handle (agent ID)" on page 180                      |
|                           | "appl_id - Application ID" on page 185                                      |
|                           | "appl_priority - Application Agent Priority" on page 195                    |
|                           | "appl_priority_type - Application Priority Type" on page 196                |
|                           | "appl_section_inserts - Section Inserts monitor element" on page 285        |
|                           | "appl_section_lookups - Section Lookups" on page 285                        |
|                           | "authority_lvl - User Authorization Level" on page 196                      |
|                           | "authority_bitmap - User Authorization Level monitor element" on page 197   |
|                           | "binds_precompiles - Binds/Precompiles Attempted" on page 372               |
|                           | "cat_cache_inserts - Catalog Cache Inserts" on page 273                     |
|                           | "cat_cache_lookups - Catalog Cache Lookups" on page 272                     |
|                           | "cat_cache_overflows - Catalog Cache Overflows" on page 274                 |
|                           | "commit_sql_stmts - Commit Statements Attempted" on page 365                |
|                           | "ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 368 |
|                           | "deadlocks - Deadlocks Detected" on page 298                                |
|                           | "direct_read_reqs - Direct Read Requests" on page 270                       |
|                           | "direct_read_time - Direct Read Time" on page 271                           |
|                           | "direct_reads - Direct Reads From Database" on page 268                     |
|                           | "direct_write_reqs - Direct Write Requests" on page 270                     |
|                           | "direct_write_time - Direct Write Time" on page 271                         |
|                           | "direct_writes - Direct Writes to Database" on page 269                     |
|                           | "disconn_time - Database Deactivation Timestamp" on page 174                |
|                           | "dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 364          |
|                           | "failed_sql_stmts - Failed Statement Operations" on page 364                |
|                           | "hash_join_overflows - Hash Join Overflows" on page 229                     |
|                           | "hash_join_small_overflows - Hash Join Small Overflows" on page 230         |
|                           | "int_auto_rebinds - Internal Automatic Rebinds" on page 369                 |
|                           | "int_commits - Internal Commits" on page 369                                |
|                           | "int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 371   |
|                           | "int_rollbacks - Internal Rollbacks" on page 370                            |
|                           | "int_rows_deleted - Internal Rows Deleted" on page 349                      |
|                           | "int_rows_inserted - Internal Rows Inserted" on page 351                    |
|                           | "int_rows_updated - Internal Rows Updated" on page 350                      |
|                           | "lock_escalation - Lock Escalation" on page 306                             |
|                           | "lock_timeouts - Number of Lock Timeouts" on page 305                       |
|                           | "lock_wait_time - Time Waited On Locks" on page 313                         |
|                           | "lock_waits - Lock Waits" on page 312                                       |
|                           | "olap_func_overflows - OLAP Function Overflows monitor element" on page 231 |
|                           | "partial_record - Partial Record monitor element" on page 408               |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| event_conn (continued)    | <p>"pkg_cache_inserts - Package Cache Inserts" on page 277</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 276</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 240</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 245</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 250</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 251</p> <p>"prefetch_wait_time - Time Waited for Prefetch" on page 264</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows" on page 282</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 284</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 283</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size" on page 282</p> <p>"rej_curs_blk - Rejected Block Cursor Requests" on page 361</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 366</p> <p>"rows_read - Rows Read" on page 348</p> <p>"rows_selected - Rows Selected" on page 346</p> <p>"int_rows_updated - Internal Rows Updated" on page 350</p> <p>"rows_written - Rows Written" on page 347</p> <p>"select_sql_stmts - Select SQL Statements Executed" on page 367</p> <p>"sequence_no - Sequence number monitor element" on page 187</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows" on page 280</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 281</p> <p>"shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 280</p> <p>"shr_workspace_size_top - Maximum Shared Workspace Size" on page 279</p> <p>"sort_overflows - Sort Overflows" on page 224</p> <p>"static_sql_stmts - Static SQL Statements Attempted" on page 363</p> <p>"system_cpu_time - System CPU Time" on page 402</p> <p>"total_hash_joins - Total Hash Joins" on page 227</p> <p>"total_hash_loops - Total Hash Loops" on page 229</p> <p>"total_olap_funcs - Total OLAP Functions monitor element" on page 230</p> |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                                     | Monitor element name                                                       |
|-------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| event_conn (continued)                                                        | "total_sec_cons - Secondary Connections" on page 214                       |
|                                                                               | "total_sort_time - Total Sort Time" on page 223                            |
|                                                                               | "total_sorts - Total Sorts" on page 222                                    |
|                                                                               | "uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 367 |
|                                                                               | "unread_prefetch_pages - Unread Prefetch Pages" on page 264                |
|                                                                               | "user_cpu_time - User CPU Time" on page 402                                |
|                                                                               | "x_lock_escals - Exclusive Lock Escalations" on page 300                   |
|                                                                               | "xquery_stmts - XQuery Statements Attempted" on page 373                   |
| event_connheader                                                              | "agent_id - Application Handle (agent ID)" on page 180                     |
|                                                                               | "appl_id - Application ID" on page 185                                     |
|                                                                               | "appl_name - Application Name" on page 185                                 |
|                                                                               | "auth_id - Authorization ID" on page 188                                   |
|                                                                               | "client_db_alias - Database Alias Used by Application" on page 190         |
|                                                                               | "client_pid - Client Process ID" on page 193                               |
|                                                                               | "client_platform - Client Operating Platform" on page 194                  |
|                                                                               | "client_prdid - Client Product/Version ID" on page 189                     |
|                                                                               | "client_protocol - Client Communication Protocol" on page 194              |
|                                                                               | "codepage_id - ID of Code Page Used by Application" on page 183            |
|                                                                               | "conn_time - Time of Database Connection" on page 174                      |
|                                                                               | "corr_token - DRDA Correlation Token" on page 193                          |
|                                                                               | "execution_id - User Login ID" on page 192                                 |
|                                                                               | "node_number - Node Number" on page 198                                    |
|                                                                               | "sequence_no - Sequence number monitor element" on page 187                |
| "territory_code - Database Territory Code" on page 195                        |                                                                            |
| event_connmemuse                                                              | "node_number - Node Number" on page 198                                    |
|                                                                               | "pool_cur_size - Current Size of Memory Pool" on page 218                  |
|                                                                               | "pool_id - Memory Pool Identifier" on page 216                             |
|                                                                               | "pool_secondary_id - Memory Pool Secondary Identifier" on page 217         |
|                                                                               | "pool_config_size - Configured Size of Memory Pool" on page 218            |
| event_data_value                                                              | "pool_watermark - Memory Pool Watermark" on page 219                       |
|                                                                               | "deadlock_id - Deadlock Event Identifier" on page 307                      |
|                                                                               | "deadlock_node - Partition Number Where Deadlock Occurred" on page 307     |
|                                                                               | "evmon_activates - Number of Event Monitor Activations" on page 410        |
|                                                                               | "participant_no - Participant within Deadlock" on page 308                 |
|                                                                               | "stmt_value_type - Value type" on page 390                                 |
|                                                                               | "stmt_history_id - Statement history identifier" on page 385               |
|                                                                               | "stmt_value_isnull - Value has null value" on page 390                     |
|                                                                               | "stmt_value_data - Value data" on page 390                                 |
|                                                                               | "stmt_value_index - Value index" on page 391                               |
| "stmt_value_isreopt - Variable used for statement reoptimization" on page 391 |                                                                            |



Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                          |
|---------------------------|-----------------------------------------------------------------------------------------------|
| event_db                  | "active_hash_joins - Active hash joins" on page 228                                           |
|                           | "appl_section_inserts - Section Inserts monitor element" on page 285                          |
|                           | "appl_section_lookups - Section Lookups" on page 285                                          |
|                           | "async_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 503 |
|                           | "binds_precompiles - Binds/Precompiles Attempted" on page 372                                 |
|                           | "blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element" on page 296      |
|                           | "cat_cache_inserts - Catalog Cache Inserts" on page 273                                       |
|                           | "cat_cache_lookups - Catalog Cache Lookups" on page 272                                       |
|                           | "cat_cache_overflows - Catalog Cache Overflows" on page 274                                   |
|                           | "cat_cache_size_top - Catalog cache high watermark" on page 275                               |
|                           | "catalog_node - Catalog Node Number" on page 176                                              |
|                           | "catalog_node_name - Catalog Node Network Name" on page 175                                   |
|                           | "commit_sql_stmts - Commit Statements Attempted" on page 365                                  |
|                           | "connections_top - Maximum Number of Concurrent Connections" on page 200                      |
|                           | "db_heap_top - Maximum Database Heap Allocated" on page 286                                   |
|                           | "ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 368                   |
|                           | "deadlocks - Deadlocks Detected" on page 298                                                  |
|                           | "direct_read_reqs - Direct Read Requests" on page 270                                         |
|                           | "direct_read_time - Direct Read Time" on page 271                                             |
|                           | "direct_reads - Direct Reads From Database" on page 268                                       |
|                           | "direct_write_reqs - Direct Write Requests" on page 270                                       |
|                           | "direct_write_time - Direct Write Time" on page 271                                           |
|                           | "direct_writes - Direct Writes to Database" on page 269                                       |
|                           | "disconn_time - Database Deactivation Timestamp" on page 174                                  |
|                           | "dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 364                            |
|                           | "evmon_activates - Number of Event Monitor Activations" on page 410                           |
|                           | "evmon_flushes - Number of Event Monitor Flushes" on page 410                                 |
|                           | "failed_sql_stmts - Failed Statement Operations" on page 364                                  |
|                           | "files_closed - Database Files Closed" on page 252                                            |
|                           | "hash_join_overflows - Hash Join Overflows" on page 229                                       |
|                           | "hash_join_small_overflows - Hash Join Small Overflows" on page 230                           |
|                           | "int_auto_rebinds - Internal Automatic Rebinds" on page 369                                   |
|                           | "int_commits - Internal Commits" on page 369                                                  |
|                           | "int_rollbacks - Internal Rollbacks" on page 370                                              |
|                           | "int_rows_deleted - Internal Rows Deleted" on page 349                                        |
|                           | "int_rows_inserted - Internal Rows Inserted" on page 351                                      |
|                           | "int_rows_updated - Internal Rows Updated" on page 350                                        |
|                           | "lock_escals - Number of Lock Escalations" on page 299                                        |
|                           | "lock_timeouts - Number of Lock Timeouts" on page 305                                         |
|                           | "lock_wait_time - Time Waited On Locks" on page 313                                           |
|                           | "lock_waits - Lock Waits" on page 312                                                         |
|                           | "log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages" on page 290      |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| event_db (continued)      | <p>"log_read_time - Log Read Time" on page 292</p> <p>"log_reads - Number of Log Pages Read" on page 288</p> <p>"log_to_redo_for_recovery - Amount of Log to be Redone for Recovery" on page 291</p> <p>"log_write_time - Log Write Time" on page 292</p> <p>"log_writes - Number of Log Pages Written" on page 288</p> <p>"num_log_read_io - Number of Log Reads" on page 293</p> <p>"num_log_write_io - Number of Log Writes" on page 292</p> <p>"num_threshold_violations - Number of threshold violations monitor element" on page 484</p> <p>"olap_func_overflows - OLAP Function Overflows monitor element" on page 231</p> <p>"partial_record - Partial Record monitor element" on page 408</p> <p>"pkg_cache_inserts - Package Cache Inserts" on page 277</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 276</p> <p>"pkg_cache_num_overflows - Package Cache Overflows" on page 278</p> <p>"pkg_cache_size_top - Package cache high watermark" on page 278</p> <p>"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258</p> <p>"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252</p> <p>"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253</p> <p>"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 259</p> <p>"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255</p> <p>"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254</p> <p>"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257</p> <p>"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 240</p> <p>"pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered" on page 261</p> <p>"pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered" on page 263</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 245</p> <p>"pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered" on page 260</p> <p>"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 262</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 250</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240</p> |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                       |
|---------------------------|------------------------------------------------------------------------------------------------------------|
| event_db (continued)      | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243                          |
|                           | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244                         |
|                           | "pool_write_time - Total Buffer Pool Physical Write Time" on page 251                                      |
|                           | "post_shrthreshold_hash_joins - Post threshold hash joins" on page 228                                     |
|                           | "post_shrthreshold_sorts - Post shared threshold sorts" on page 221                                        |
|                           | "prefetch_wait_time - Time Waited for Prefetch" on page 264                                                |
|                           | "priv_workspace_num_overflows - Private Workspace Overflows" on page 282                                   |
|                           | "priv_workspace_section_inserts - Private Workspace Section Inserts" on page 284                           |
|                           | "priv_workspace_section_lookups - Private Workspace Section Lookups" on page 283                           |
|                           | "priv_workspace_size_top - Maximum Private Workspace Size" on page 282                                     |
|                           | "rollback_sql_stmts - Rollback Statements Attempted" on page 366                                           |
|                           | "rows_deleted - Rows Deleted" on page 345                                                                  |
|                           | "rows_inserted - Rows Inserted" on page 345                                                                |
|                           | "rows_read - Rows Read" on page 348                                                                        |
|                           | "rows_selected - Rows Selected" on page 346                                                                |
|                           | "rows_updated - Rows Updated" on page 346                                                                  |
|                           | "sec_log_used_top - Maximum Secondary Log Space Used" on page 286                                          |
|                           | "select_sql_stmts - Select SQL Statements Executed" on page 367                                            |
|                           | "server_platform - Server Operating System" on page 171                                                    |
|                           | "shr_workspace_num_overflows - Shared Workspace Overflows" on page 280                                     |
|                           | "shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 281                             |
|                           | "shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 280                             |
|                           | "shr_workspace_size_top - Maximum Shared Workspace Size" on page 279                                       |
|                           | "sort_overflows - Sort Overflows" on page 224                                                              |
|                           | "static_sql_stmts - Static SQL Statements Attempted" on page 363                                           |
|                           | "stats_cache_size - Size of statistics cache monitor element" on page 501                                  |
|                           | "stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 504 |
|                           | "stats_fabrications - Total number of statistics fabrications monitor elements" on page 502                |
|                           | "sync_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 502              |
|                           | "sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 504     |
|                           | "tot_log_used_top - Maximum Total Log Space Used" on page 287                                              |
|                           | "total_cons - Connects Since Database Activation" on page 209                                              |
|                           | "total_hash_joins - Total Hash Joins" on page 227                                                          |
|                           | "total_hash_loops - Total Hash Loops" on page 229                                                          |
|                           | "total_olap_funcs - Total OLAP Functions monitor element" on page 230                                      |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                   | Monitor element name                                                           |
|---------------------------------------------|--------------------------------------------------------------------------------|
| event_db (continued)                        | "total_sort_time - Total Sort Time" on page 223                                |
|                                             | "total_sorts - Total Sorts" on page 222                                        |
|                                             | "uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 367     |
|                                             | "unread_prefetch_pages - Unread Prefetch Pages" on page 264                    |
|                                             | "x_lock_escalations - Exclusive Lock Escalations" on page 300                  |
|                                             | "xquery_stmts - XQuery Statements Attempted" on page 373                       |
| event_dbheader                              | "conn_time - Time of Database Connection" on page 174                          |
|                                             | "db_name - Database Name" on page 172                                          |
|                                             | "db_path - Database Path" on page 173                                          |
| event_dbmemuse                              | "node_number - Node Number" on page 198                                        |
|                                             | "pool_cur_size - Current Size of Memory Pool" on page 218                      |
|                                             | "pool_id - Memory Pool Identifier" on page 216                                 |
|                                             | "pool_config_size - Configured Size of Memory Pool" on page 218                |
|                                             | "pool_watermark - Memory Pool Watermark" on page 219                           |
| event_deadlock                              | "deadlock_id - Deadlock Event Identifier" on page 307                          |
|                                             | "deadlock_node - Partition Number Where Deadlock Occurred" on page 307         |
|                                             | "dl_conns - Connections Involved in Deadlock" on page 306                      |
|                                             | "evmon_activates - Number of Event Monitor Activations" on page 410            |
|                                             | "rolled_back_agent_id - Rolled Back Agent" on page 318                         |
|                                             | "rolled_back_appl_id - Rolled Back Application" on page 317                    |
|                                             | "rolled_back_participant_no - Rolled Back Application Participant" on page 308 |
|                                             | "rolled_back_sequence_no - Rolled Back Sequence Number" on page 318            |
| "start_time - Event Start Time" on page 379 |                                                                                |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                       |
|---------------------------|------------------------------------------------------------------------------------------------------------|
| event_detailed_dlconn     | "agent_id - Application Handle (agent ID)" on page 180                                                     |
|                           | "appl_id - Application ID" on page 185                                                                     |
|                           | "appl_id_holding_lk - Application ID Holding Lock" on page 316                                             |
|                           | "blocking_cursor - Blocking Cursor" on page 458                                                            |
|                           | "consistency_token - Package Consistency Token" on page 376                                                |
|                           | "creator - Application Creator" on page 378                                                                |
|                           | "cursor_name - Cursor Name" on page 377                                                                    |
|                           | "deadlock_id - Deadlock Event Identifier" on page 307                                                      |
|                           | "deadlock_node - Partition Number Where Deadlock Occurred" on page 307                                     |
|                           | "data_partition_id - Data partition identifier monitor element" on page 298                                |
|                           | "evmon_activates - Number of Event Monitor Activations" on page 410                                        |
|                           | "lock_escalation - Lock Escalation" on page 306                                                            |
|                           | "lock_mode - Lock Mode" on page 301                                                                        |
|                           | "lock_mode_requested - Lock Mode Requested" on page 306                                                    |
|                           | "lock_node - Lock Node" on page 304                                                                        |
|                           | "lock_object_name - Lock Object Name" on page 304                                                          |
|                           | "lock_object_type - Lock Object Type Waited On" on page 303                                                |
|                           | "lock_wait_start_time - Lock Wait Start Timestamp" on page 315                                             |
|                           | "locks_held - Locks Held" on page 297                                                                      |
|                           | "locks_in_list - Number of Locks Reported" on page 309                                                     |
|                           | "package_name - Package Name" on page 375                                                                  |
|                           | "package_version_id - Package Version" on page 376                                                         |
|                           | "participant_no - Participant within Deadlock" on page 308                                                 |
|                           | "participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application" on page 308 |
|                           | "section_number - Section Number" on page 377                                                              |
|                           | "sequence_no - Sequence number monitor element" on page 187                                                |
|                           | "sequence_no_holding_lk - Sequence Number Holding Lock" on page 317                                        |
|                           | "start_time - Event Start Time" on page 379                                                                |
|                           | "stmt_operation/operation - Statement Operation" on page 374                                               |
|                           | "stmt_text - SQL Statement Text monitor element" on page 380                                               |
|                           | "stmt_type - Statement Type" on page 373                                                                   |
|                           | "table_name - Table Name" on page 343                                                                      |
|                           | "table_schema - Table Schema Name" on page 344                                                             |
|                           | "tablespace_name - Table Space Name" on page 320                                                           |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                   | Monitor element name                                                                                       |
|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| event_dlconn                                                | "agent_id - Application Handle (agent ID)" on page 180                                                     |
|                                                             | "appl_id - Application ID" on page 185                                                                     |
|                                                             | "appl_id_holding_lk - Application ID Holding Lock" on page 316                                             |
|                                                             | "deadlock_id - Deadlock Event Identifier" on page 307                                                      |
|                                                             | "data_partition_id - Data partition identifier monitor element" on page 298                                |
|                                                             | "deadlock_node - Partition Number Where Deadlock Occurred" on page 307                                     |
|                                                             | "evmon_activates - Number of Event Monitor Activations" on page 410                                        |
|                                                             | "lock_attributes - Lock Attributes" on page 309                                                            |
|                                                             | "lock_count - Lock Count" on page 311                                                                      |
|                                                             | "lock_current_mode - Original Lock Mode Before Conversion" on page 312                                     |
|                                                             | "lock_escalation - Lock Escalation" on page 306                                                            |
|                                                             | "lock_hold_count - Lock Hold Count" on page 311                                                            |
|                                                             | "lock_mode - Lock Mode" on page 301                                                                        |
|                                                             | "lock_mode_requested - Lock Mode Requested" on page 306                                                    |
|                                                             | "lock_name - Lock Name" on page 309                                                                        |
|                                                             | "lock_node - Lock Node" on page 304                                                                        |
|                                                             | "lock_object_name - Lock Object Name" on page 304                                                          |
|                                                             | "lock_object_type - Lock Object Type Waited On" on page 303                                                |
|                                                             | "lock_release_flags - Lock Release Flags" on page 310                                                      |
|                                                             | "lock_wait_start_time - Lock Wait Start Timestamp" on page 315                                             |
|                                                             | "participant_no - Participant within Deadlock" on page 308                                                 |
|                                                             | "participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application" on page 308 |
|                                                             | "sequence_no - Sequence number monitor element" on page 187                                                |
|                                                             | "sequence_no_holding_lk - Sequence Number Holding Lock" on page 317                                        |
|                                                             | "start_time - Event Start Time" on page 379                                                                |
|                                                             | "table_name - Table Name" on page 343                                                                      |
|                                                             | "table_schema - Table Schema Name" on page 344                                                             |
| "tablespace_name - Table Space Name" on page 320            |                                                                                                            |
| event_histogrambin                                          | "statistics_timestamp - Statistics timestamp monitor element" on page 492                                  |
|                                                             | "bin_id - Histogram bin identifier monitor element" on page 473                                            |
|                                                             | "histogram_type - Histogram type monitor element" on page 483                                              |
|                                                             | "service_class_id - Service class ID monitor element" on page 490                                          |
|                                                             | "work_action_set_id - Work action set ID monitor element" on page 498                                      |
|                                                             | "work_class_id - Work class ID monitor element" on page 499                                                |
|                                                             | "top - Histogram bin top monitor element" on page 497                                                      |
|                                                             | "bottom - Histogram bin bottom monitor element" on page 474                                                |
| "number_in_bin - Number in bin monitor element" on page 485 |                                                                                                            |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                         | Monitor element name                                                            |
|-------------------------------------------------------------------|---------------------------------------------------------------------------------|
| event_log_header                                                  | "byte_order - Byte Order of Event Data" on page 407                             |
|                                                                   | "codepage_id - ID of Code Page Used by Application" on page 183                 |
|                                                                   | "event_monitor_name - Event Monitor Name" on page 408                           |
|                                                                   | "num_nodes_in_db2_instance - Number of Nodes in Partition" on page 406          |
|                                                                   | "server_prdid - Server Product/Version ID" on page 169                          |
|                                                                   | "server_instance_name - Server Instance Name" on page 168                       |
|                                                                   | "territory_code - Database Territory Code" on page 195                          |
|                                                                   | "version - Version of Monitor Data" on page 408                                 |
| event_overflow                                                    | "count - Number of Event Monitor Overflows" on page 406                         |
|                                                                   | "first_overflow_time - Time of First Event Overflow" on page 407                |
|                                                                   | "last_overflow_time - Time of Last Event Overflow" on page 407                  |
|                                                                   | "node_number - Node Number" on page 198                                         |
| event_qstats                                                      | "last_wlm_reset - Time of last reset monitor element" on page 484               |
|                                                                   | "service_superclass_name - Service superclass name monitor element" on page 491 |
|                                                                   | "service_subclass_name - Service subclass name monitor element" on page 491     |
|                                                                   | "statistics_timestamp - Statistics timestamp monitor element" on page 492       |
|                                                                   | "threshold_domain - Threshold domain monitor element" on page 493               |
|                                                                   | "threshold_name - Threshold name monitor element" on page 494                   |
|                                                                   | "threshold_predicate - Threshold predicate monitor element" on page 494         |
|                                                                   | "thresholdid - Threshold ID monitor element" on page 495                        |
|                                                                   | "work_action_set_name - Work action set name monitor element" on page 499       |
|                                                                   | "work_class_name - Work class name monitor element" on page 499                 |
|                                                                   | "queue_assignments_total - Queue assignments total monitor element" on page 486 |
|                                                                   | "queue_size_top - Queue size top monitor element" on page 487                   |
| "queue_time_total - Queue time total monitor element" on page 487 |                                                                                 |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                      |
|---------------------------|-----------------------------------------------------------------------------------------------------------|
| event_scstats             | "statistics_timestamp - Statistics timestamp monitor element" on page 492                                 |
|                           | "service_superclass_name - Service superclass name monitor element" on page 491                           |
|                           | "service_subclass_name - Service subclass name monitor element" on page 491                               |
|                           | "service_class_id - Service class ID monitor element" on page 490                                         |
|                           | "last_wlm_reset - Time of last reset monitor element" on page 484                                         |
|                           | "coord_act_completed_total - Coordinator activities completed total monitor element" on page 476          |
|                           | "coord_act_aborted_total - Coordinator activities aborted total monitor element" on page 476              |
|                           | "coord_act_rejected_total - Coordinator activities rejected total monitor element" on page 477            |
|                           | "coord_act_lifetime_top - Coordinator activity lifetime top monitor element" on page 477                  |
|                           | "concurrent_connection_top - Concurrent connection top monitor element" on page 475                       |
|                           | "concurrent_act_top - Concurrent activity top monitor element" on page 474                                |
|                           | "cost_estimate_top - Cost estimate top monitor element" on page 478                                       |
|                           | "rows_returned_top - Actual rows returned top monitor element" on page 489                                |
|                           | "temp_tablespace_top - Temporary table space top monitor element" on page 492                             |
|                           | "coord_act_lifetime_avg - Coordinator activity lifetime average monitor element" on page 478              |
|                           | "coord_act_queue_time_avg - Coordinator activity queue time average monitor element" on page 479          |
|                           | "coord_act_exec_time_avg - Coordinator activities execution time average monitor element" on page 480     |
|                           | "request_exec_time_avg - Request execution time average monitor element" on page 480                      |
|                           | "coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element" on page 481        |
|                           | "coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element" on page 482 |
| event_start               | "start_time - Event Start Time" on page 379                                                               |



Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                       |
|---------------------------|------------------------------------------------------------------------------------------------------------|
| event_stmt                | "agent_id - Application Handle (agent ID)" on page 180                                                     |
|                           | "agents_top - Number of Agents Created" on page 399                                                        |
|                           | "appl_id - Application ID" on page 185                                                                     |
|                           | "blocking_cursor - Blocking Cursor" on page 458                                                            |
|                           | "consistency_token - Package Consistency Token" on page 376                                                |
|                           | "creator - Application Creator" on page 378                                                                |
|                           | "cursor_name - Cursor Name" on page 377                                                                    |
|                           | "fetch_count - Number of Successful Fetches" on page 382                                                   |
|                           | "int_rows_deleted - Internal Rows Deleted" on page 349                                                     |
|                           | "int_rows_inserted - Internal Rows Inserted" on page 351                                                   |
|                           | "int_rows_updated - Internal Rows Updated" on page 350                                                     |
|                           | "package_name - Package Name" on page 375                                                                  |
|                           | "package_version_id - Package Version" on page 376                                                         |
|                           | "partial_record - Partial Record monitor element" on page 408                                              |
|                           | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                                           |
|                           | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                                          |
|                           | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                                         |
|                           | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                                        |
|                           | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238                            |
|                           | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240                           |
|                           | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243                          |
|                           | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244                         |
|                           | "rows_read - Rows Read" on page 348                                                                        |
|                           | "rows_written - Rows Written" on page 347                                                                  |
|                           | "section_number - Section Number" on page 377                                                              |
|                           | "sequence_no - Sequence number monitor element" on page 187                                                |
|                           | "sort_overflows - Sort Overflows" on page 224                                                              |
|                           | "sql_req_id - Request Identifier for SQL Statement" on page 411                                            |
|                           | "sqlca - SQL Communications Area (SQLCA)" on page 383                                                      |
|                           | "start_time - Event Start Time" on page 379                                                                |
|                           | "stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 504 |
|                           | "stmt_operation/operation - Statement Operation" on page 374                                               |
|                           | "stmt_text - SQL Statement Text monitor element" on page 380                                               |
|                           | "stmt_type - Statement Type" on page 373                                                                   |
|                           | "stop_time - Event Stop Time" on page 379                                                                  |
|                           | "sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 504     |
|                           | "system_cpu_time - System CPU Time" on page 402                                                            |
|                           | "total_sort_time - Total Sort Time" on page 223                                                            |
|                           | "total_sorts - Total Sorts" on page 222                                                                    |
|                           | "user_cpu_time - User CPU Time" on page 402                                                                |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                           | Monitor element name                                                              |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| event_stmt_history                                                  | "evmon_activates - Number of Event Monitor Activations" on page 410               |
|                                                                     | "deadlock_id - Deadlock Event Identifier" on page 307                             |
|                                                                     | "deadlock_node - Partition Number Where Deadlock Occurred" on page 307            |
|                                                                     | "participant_no - Participant within Deadlock" on page 308                        |
|                                                                     | "stmt_history_id - Statement history identifier" on page 385                      |
|                                                                     | "stmt_first_use_time - Statement first use time" on page 385                      |
|                                                                     | "stmt_last_use_time - Statement last use time monitor element" on page 385        |
|                                                                     | "stmt_lock_timeout - Statement lock timeout" on page 386                          |
|                                                                     | "stmt_isolation - Statement isolation" on page 386                                |
|                                                                     | "package_name - Package Name" on page 375                                         |
|                                                                     | "package_version_id - Package Version" on page 376                                |
|                                                                     | "section_number - Section Number" on page 377                                     |
|                                                                     | "creator - Application Creator" on page 378                                       |
|                                                                     | "stmt_type - Statement Type" on page 373                                          |
|                                                                     | "stmt_text - SQL Statement Text monitor element" on page 380                      |
|                                                                     | "comp_env_desc - Compilation environment handle" on page 389                      |
|                                                                     | "stmt_nest_level - Statement nesting level" on page 387                           |
|                                                                     | "stmt_invocation_id - Statement invocation identifier" on page 387                |
|                                                                     | "stmt_query_id - Statement query identifier" on page 388                          |
|                                                                     | "stmt_source_id - Statement source identifier" on page 388                        |
|                                                                     | "sequence_no - Sequence number monitor element" on page 187                       |
| "stmt_pkgcache_id - Statement package cache identifier" on page 389 |                                                                                   |
| "stmt_text - SQL Statement Text monitor element" on page 380        |                                                                                   |
| event_subsection                                                    | "agent_id - Application Handle (agent ID)" on page 180                            |
|                                                                     | "num_agents - Number of Agents Working on a Statement" on page 399                |
|                                                                     | "partial_record - Partial Record monitor element" on page 408                     |
|                                                                     | "ss_exec_time - Subsection Execution Elapsed Time" on page 393                    |
|                                                                     | "ss_node_number - Subsection Node Number" on page 392                             |
|                                                                     | "ss_number - Subsection Number" on page 392                                       |
|                                                                     | "ss_sys_cpu_time - System CPU Time used by Subsection" on page 403                |
|                                                                     | "ss_usr_cpu_time - User CPU Time used by Subsection" on page 403                  |
|                                                                     | "tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows" on page 396 |
|                                                                     | "tq_rows_read - Number of Rows Read from Tablequeues" on page 395                 |
|                                                                     | "tq_rows_written - Number of Rows Written to Tablequeues" on page 396             |
|                                                                     | "tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed" on page 394  |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| event_table               | <p>“data_object_pages - Data Object Pages” on page 352</p> <p>“event_time - Event Time” on page 409</p> <p>“evmon_activates - Number of Event Monitor Activations” on page 410</p> <p>“evmon_flushes - Number of Event Monitor Flushes” on page 410</p> <p>“index_object_pages - Index Object Pages” on page 353</p> <p>“lob_object_pages - LOB Object Pages” on page 353</p> <p>“long_object_pages - Long Object Pages” on page 354</p> <p>“overflow_accesses - Accesses to Overflowed Records” on page 349</p> <p>“page_reorgs - Page Reorganizations” on page 352</p> <p>“partial_record - Partial Record monitor element” on page 408</p> <p>“rows_read - Rows Read” on page 348</p> <p>“rows_written - Rows Written” on page 347</p> <p>“table_name - Table Name” on page 343</p> <p>“table_schema - Table Schema Name” on page 344</p> <p>“table_type - Table Type” on page 342</p> <p>“data_partition_id - Data partition identifier monitor element” on page 298</p> |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                      |
|---------------------------|-------------------------------------------------------------------------------------------|
| event_tablespace          | "direct_read_reqs - Direct Read Requests" on page 270                                     |
|                           | "direct_read_time - Direct Read Time" on page 271                                         |
|                           | "direct_reads - Direct Reads From Database" on page 268                                   |
|                           | "direct_write_reqs - Direct Write Requests" on page 270                                   |
|                           | "direct_write_time - Direct Write Time" on page 271                                       |
|                           | "direct_writes - Direct Writes to Database" on page 269                                   |
|                           | "event_time - Event Time" on page 409                                                     |
|                           | "evmon_activates - Number of Event Monitor Activations" on page 410                       |
|                           | "evmon_flushes - Number of Event Monitor Flushes" on page 410                             |
|                           | "files_closed - Database Files Closed" on page 252                                        |
|                           | "partial_record - Partial Record monitor element" on page 408                             |
|                           | "pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 258          |
|                           | "pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 252 |
|                           | "pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 253               |
|                           | "pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 259   |
|                           | "pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 255               |
|                           | "pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 254             |
|                           | "pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 257                   |
|                           | "pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 257                 |
|                           | "pool_data_l_reads - Buffer Pool Data Logical Reads" on page 237                          |
|                           | "pool_data_p_reads - Buffer Pool Data Physical Reads" on page 239                         |
|                           | "pool_data_writes - Buffer Pool Data Writes" on page 240                                  |
|                           | "pool_index_l_reads - Buffer Pool Index Logical Reads" on page 242                        |
|                           | "pool_index_p_reads - Buffer Pool Index Physical Reads" on page 243                       |
|                           | "pool_index_writes - Buffer Pool Index Writes" on page 245                                |
|                           | "pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 262                       |
|                           | "pool_read_time - Total Buffer Pool Physical Read Time" on page 250                       |
|                           | "pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 238           |
|                           | "pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 240          |
|                           | "pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 243         |
|                           | "pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 244        |
|                           | "pool_write_time - Total Buffer Pool Physical Write Time" on page 251                     |
|                           | "tablespace_name - Table Space Name" on page 320                                          |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups | Monitor element name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| event_thresholdviolations | <p>“activate_timestamp - Activate timestamp monitor element” on page 470</p> <p>“coord_partition_num - Coordinator partition number monitor element” on page 478</p> <p>“thresholdid - Threshold ID monitor element” on page 495</p> <p>“threshold_predicate - Threshold predicate monitor element” on page 494</p> <p>“threshold_action - Threshold action monitor element” on page 493</p> <p>“threshold_maxvalue - Threshold maximum value monitor element” on page 494</p> <p>“threshold_queuesize - Threshold queue size monitor element” on page 495</p> <p>“time_of_violation - Time of violation monitor element” on page 496</p> <p>“agent_id - Application Handle (agent ID)” on page 180</p> <p>“appl_id - Application ID” on page 185</p> <p>“uow_id - Unit of work ID monitor element” on page 497</p> <p>“activity_id - Activity ID monitor element” on page 470</p> <p>“activity_collected - Activity collected monitor element” on page 470</p> |
| event_wlstats             | <p>“statistics_timestamp - Statistics timestamp monitor element” on page 492</p> <p>“last_wlm_reset - Time of last reset monitor element” on page 484</p> <p>“workload_id - Workload ID monitor element” on page 500</p> <p>“workload_name - Workload name monitor element” on page 500</p> <p>“concurrent_wlo_top - Concurrent workload occurrences top monitor element” on page 475</p> <p>“concurrent_wlo_act_top - Concurrent WLO activity top monitor element” on page 475</p> <p>“coord_act_rejected_total - Coordinator activities rejected total monitor element” on page 477</p> <p>“coord_act_aborted_total - Coordinator activities aborted total monitor element” on page 476</p> <p>“coord_act_completed_total - Coordinator activities completed total monitor element” on page 476</p> <p>“wlo_completed_total - Workload occurrences completed total monitor element” on page 498</p>                                                           |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                | Monitor element name                                                                                      |
|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| event_wcstats                                            | "act_total - Activities total monitor element" on page 473                                                |
|                                                          | "coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element" on page 481        |
|                                                          | "coord_act_exec_time_avg - Coordinator activities execution time average monitor element" on page 480     |
|                                                          | "coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element" on page 482 |
|                                                          | "coord_act_lifetime_avg - Coordinator activity lifetime average monitor element" on page 478              |
|                                                          | "coord_act_lifetime_top - Coordinator activity lifetime top monitor element" on page 477                  |
|                                                          | "coord_act_queue_time_avg - Coordinator activity queue time average monitor element" on page 479          |
|                                                          | "cost_estimate_top - Cost estimate top monitor element" on page 478                                       |
|                                                          | "last_wlm_reset - Time of last reset monitor element" on page 484                                         |
|                                                          | "rows_returned_top - Actual rows returned top monitor element" on page 489                                |
|                                                          | "statistics_timestamp - Statistics timestamp monitor element" on page 492                                 |
|                                                          | "temp_tablespace_top - Temporary table space top monitor element" on page 492                             |
|                                                          | "work_action_set_id - Work action set ID monitor element" on page 498                                     |
|                                                          | "work_action_set_name - Work action set name monitor element" on page 499                                 |
|                                                          | "work_class_id - Work class ID monitor element" on page 499                                               |
|                                                          | "work_class_name - Work class name monitor element" on page 499                                           |
| event_xact                                               | "agent_id - Application Handle (agent ID)" on page 180                                                    |
|                                                          | "appl_id - Application ID" on page 185                                                                    |
|                                                          | "lock_escals - Number of Lock Escalations" on page 299                                                    |
|                                                          | "lock_wait_time - Time Waited On Locks" on page 313                                                       |
|                                                          | "locks_held_top - Maximum Number of Locks Held" on page 305                                               |
|                                                          | "partial_record - Partial Record monitor element" on page 408                                             |
|                                                          | "prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 201                             |
|                                                          | "rows_read - Rows Read" on page 348                                                                       |
|                                                          | "rows_written - Rows Written" on page 347                                                                 |
|                                                          | "sequence_no - Sequence number monitor element" on page 187                                               |
|                                                          | "system_cpu_time - System CPU Time" on page 402                                                           |
|                                                          | "uow_log_space_used - Unit of Work Log Space Used" on page 289                                            |
|                                                          | "uow_start_time - Unit of Work Start Timestamp" on page 201                                               |
|                                                          | "uow_status - Unit of Work Status" on page 203                                                            |
|                                                          | "uow_stop_time - Unit of Work Stop Timestamp" on page 202                                                 |
|                                                          | "user_cpu_time - User CPU Time" on page 402                                                               |
| "x_lock_escals - Exclusive Lock Escalations" on page 300 |                                                                                                           |

Table 20. Event Monitor Logical Data Groups and Monitor Elements (continued)

| Event logical data groups                                                   | Monitor element name                                                   |
|-----------------------------------------------------------------------------|------------------------------------------------------------------------|
| lock                                                                        | "lock_attributes - Lock Attributes" on page 309                        |
|                                                                             | "lock_count - Lock Count" on page 311                                  |
|                                                                             | "lock_current_mode - Original Lock Mode Before Conversion" on page 312 |
|                                                                             | "lock_escalation - Lock Escalation" on page 306                        |
|                                                                             | "lock_hold_count - Lock Hold Count" on page 311                        |
|                                                                             | "lock_mode - Lock Mode" on page 301                                    |
|                                                                             | "lock_name - Lock Name" on page 309                                    |
|                                                                             | "lock_object_name - Lock Object Name" on page 304                      |
|                                                                             | "lock_object_type - Lock Object Type Waited On" on page 303            |
|                                                                             | "lock_release_flags - Lock Release Flags" on page 310                  |
|                                                                             | "lock_status - Lock Status" on page 302                                |
|                                                                             | "node_number - Node Number" on page 198                                |
|                                                                             | "table_file_id - Table File ID" on page 351                            |
|                                                                             | "table_name - Table Name" on page 343                                  |
|                                                                             | "table_schema - Table Schema Name" on page 344                         |
| "tablespace_name - Table Space Name" on page 320                            |                                                                        |
| "data_partition_id - Data partition identifier monitor element" on page 298 |                                                                        |
| sqlca                                                                       | sqlcab                                                                 |
|                                                                             | sqlcode                                                                |
|                                                                             | sqlerrml                                                               |
|                                                                             | sqlcaid                                                                |
|                                                                             | sqlerrmc                                                               |
|                                                                             | sqlerrp                                                                |
|                                                                             | sqlerrd                                                                |
|                                                                             | sqlwarn                                                                |
|                                                                             | sqlstate                                                               |

## 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 21. COLLECT ACTIVITY DATA settings

| Setting for COLLECT ACTIVITY DATA | Logical data groups collected                              |
|-----------------------------------|------------------------------------------------------------|
| NONE                              | none                                                       |
| WITHOUT DETAILS                   | event_activity                                             |
| WITH DETAILS                      | event_activity<br>event_activitystmt                       |
| WITH DETAILS AND VALUES           | event_activity<br>event_activitystmt<br>event_activityvals |





---

## Chapter 9. Database system monitor elements

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 on **DB2 Connect** applications. Including information on DCS applications running at the gateway, SQL statements being executed, and database connections.
- Information on **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.

### 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 uppercased 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 on how you can use the information collected by the monitor element when monitoring your database system.

## Server identification and status monitor elements

The following elements provide identification and status information about the server.

### 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 22. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

**Usage** This element may 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.

### 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 23. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

Table 24. Event Monitoring Information

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

**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\_db2\_type - Database Manager Type at Monitored (Server) Node

Identifies the type of database manager being monitored.

**Element identifier**  
server\_db2\_type

**Element type**  
information

Table 25. 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\_prdid - Server Product/Version ID

The product and version that is running on the server.

**Element identifier**  
server\_prdid

**Element type**  
information

Table 26. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | collected             | Basic          |

Table 27. 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.

**Element identifier**

server\_version

**Element type**

information

*Table 28. 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\_level - Service Level

This is the current corrective service level of the DB2 instance.

**Element identifier**

service\_level

**Element type**

information

Table 29. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

## server\_platform - Server Operating System

The operating system running the database server.

**Element identifier**

server\_platform

**Element type**

information

Table 30. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 31. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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*.

## product\_name - Product Name

Details of the version of the DB2 instance that is running.

**Element identifier**

product\_name

**Element type**

information

Table 32. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

## db2\_status - Status of DB2 Instance

The current status of the instance of the database manager.

**Element identifier**

db2\_status

**Element type**

information

Table 33. 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.                                                                                                                                   |

## time\_zone\_disp - Time Zone Displacement

Number of seconds that the local time zone is displaced from Greenwich Mean Time (GMT).

### Element identifier

time\_zone\_disp

### Element type

information

Table 34. 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.

## Database identification and status monitor elements

The following elements provide identification and status information about the database.

### db\_name - Database Name

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.

### Element identifier

db\_name

### Element type

information

Table 35. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Database       | dbase_remote          | Basic          |

Table 35. Snapshot Monitoring Information (continued)

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| 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 36. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_dbheader        | -              |

**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.

## db\_path - Database Path

The full path of the location where the database is stored on the monitored system.

### Element identifier

db\_path

### Element type

information

Table 37. 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 38. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_dbheader        | -              |

**Usage** This element can be used with the *db\_name* monitor element to identify the specific database to which the data applies.

## db\_conn\_time - Database Activation Timestamp

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.

**Element identifier**

db\_conn\_time

**Element type**

timestamp

*Table 39. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Database       | dbase                 | Timestamp              |
| Table Space    | tablespace_list       | Buffer Pool, Timestamp |
| Table          | table_list            | Timestamp              |

**Usage** Use this element with the *disconn\_time* monitor element to calculate the total connection time.

## conn\_time - Time of Database Connection

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.

**Element identifier**

conn\_time

**Element type**

timestamp

*Table 40. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_dbheader        | -              |
| Connections | event_connheader      | -              |

**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).

## 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 41. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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).

## db\_status - Status of Database

The current status of the database.

**Element identifier**

db\_status

**Element type**

information

Table 42. 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.                                                                                                                                                                                                               |

## catalog\_node\_name - Catalog Node Network Name

The network name of the catalog node.

**Element identifier**

catalog\_node\_name

**Element type**

information

Table 43. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 44. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**Usage** Use this element to determine the location of a database.

## db\_location - Database Location

The location of the database in relation to the application.

**Element identifier**

db\_location

**Element type**

information

Table 45. 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

## 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 46. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 47. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

## last\_backup - Last Backup Timestamp

The date and time that the latest database backup was completed.

**Element identifier**  
last\_backup

**Element type**  
timestamp

*Table 48. 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.

## **db\_storage\_path - Automatic storage path**

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.

**Element identifier**  
db\_storage\_path

**Element type**  
information

*Table 49. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | db_sto_path_info      | Basic          |

**Usage** You can use this element with the num\_db\_storage\_paths monitor element to identify the storage paths that are associated with this database.

## **num\_db\_storage\_paths - Number of automatic storage paths**

This element shows the number of automatic storage paths associated with this database.

**Element identifier**  
num\_db\_storage\_paths

**Element type**  
information

*Table 50. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

**Usage** You can use this element with the db\_storage\_path monitor element to identify the storage paths that are associated with this database.

## **sto\_path\_free\_sz - Automatic Storage Path Free Space**

This element shows the amount of free space 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.

**Element identifier**  
fs\_free\_size

**Element type**  
information

*Table 51. 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
- fs\_type

## fs\_used\_size - Amount of Space Used on a File System

This element shows the amount of space already used on a file system pointed to by a storage path.

**Element identifier**  
fs\_used\_size

**Element type**  
information

*Table 52. 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
- fs\_type

## fs\_total\_size - Total Size of a File System

This element shows the capacity of a file system pointed to by a storage path.

**Element identifier**  
fs\_total\_size

**Element type**  
information

*Table 53. 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\_type

## fs\_id - Unique File System Identification Number

This element shows the unique identification number provided by the operating system for a file system pointed to by a storage path.

**Element identifier**

fs\_id

**Element type**

information

*Table 54. 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\_total\_size
- fs\_type

## fs\_type - File System Type

This element shows the type of a file system that is pointed to by a storage path. This file system type is provided by the operating system..

**Element identifier**

fs\_type

**Element type**

information

*Table 55. 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\_total\_size
- fs\_id

---

## Application identification and status monitor elements

The following elements provide information about databases and their related applications.

### agent\_id - Application Handle (agent ID)

A system-wide unique ID for the application. On a single-partitioned database, this identifier consists of a 16 bit counter. On a multi-partitioned database, this identifier consists of the coordinating partition number concatenated with a 16 bit counter. In addition, this identifier will be the same on every partition where the application may make a secondary connection.

#### Element identifier

agent\_id

#### Element type

information

Table 56. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| Lock            | appl_lock_list        | Basic          |
| DCS Application | dcx_appl_info         | Basic          |

Table 57. Event Monitoring Information

| Event Type             | Logical Data Grouping     | Monitor Switch |
|------------------------|---------------------------|----------------|
| Connections            | event_connheader          | -              |
| Statements             | event_stmt                | -              |
| Statements             | event_subsection          | -              |
| Deadlocks              | event_dlconn              | -              |
| Deadlocks with Details | event_detailed_dlconn     | -              |
| Threshold violations   | event_thresholdviolations | -              |
| Activities             | event_activity            | -              |

### Usage

The application handle, also known as 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 back-level compatibility, 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.

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.

## appl\_status - Application Status

The current status of the application.

**Element identifier**  
appl\_status

**Element type**  
information

*Table 58. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_id_info          | Basic          |
| Lock           | appl_lock_list        | Basic          |

*Table 59. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**Usage** This element can help you diagnose potential application problems. Values for this field are:

| API Constant      | Description                                                                                                                                                                                        |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLM_CONNECTPEND  | <b>Database Connect Pending:</b> The application has initiated a database connection but the request has not yet completed.                                                                        |
| SQLM_CONNECTED    | <b>Database Connect Completed:</b> The application has initiated a database connection and the request has completed.                                                                              |
| SQLM_UOWEXEC      | <b>Unit of Work Executing:</b> The database manager is executing requests on behalf of the unit of work.                                                                                           |
| 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_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_COMMIT_ACT   | <b>Commit Active:</b> The unit of work is committing its database changes.                                                                                                                         |
| SQLM_ROLLBACK_ACT | <b>Rollback Active:</b> The unit of work is rolling back its database changes.                                                                                                                     |
| SQLM_RECOMP       | <b>Recompiling:</b> The database manager is recompiling (that is, rebinding) a plan on behalf of the application.                                                                                  |

| 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_INTR           | <b>Request Interrupted:</b> An interrupt of a request is in progress.                                                                                                                                                                                                                                                                                                                                                                |
| 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_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_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_THCOMT         | <b>Transaction Heuristically Committed:</b> The unit of work is part of a global transaction that has been heuristically committed.                                                                                                                                                                                                                                                                                                  |
| 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_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_CREATE_DB      | <b>Creating Database:</b> The agent has initiated a request to create a database and that request has not yet completed.                                                                                                                                                                                                                                                                                                             |
| 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.                                                                                                                                                                                                                                                                                                                                              |
| SQLM_BACKUP         | <b>Backing Up Database:</b> The application is performing a backup of the database.                                                                                                                                                                                                                                                                                                                                                  |
| SQLM_LOAD           | <b>Data Fast Load:</b> The application is performing a "fast load" of data into the database.                                                                                                                                                                                                                                                                                                                                        |
| SQLM_UNLOAD         | <b>Data Fast Unload:</b> The application is performing a "fast unload" of data from the database.                                                                                                                                                                                                                                                                                                                                    |
| 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.                                                                                                                                                             |



| API Constant               | Description                                                                                                                          |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| SQLM_QUIESCE_TABLESPACE    | <b>Quiescing a Table space:</b> The application is performing a quiesce table space request.                                         |
| SQLM_WAITFOR_REMOTE        | <b>Pending remote request:</b> The application is waiting for a response from a remote partition in a partitioned database instance. |
| SQLM_REMOTE_RQST           | <b>Federated request pending:</b> The application is waiting for results from a federated data source.                               |
| SQLM_ROLLBACK_TO_SAVEPOINT | <b>Rollback to savepoint:</b> The application is rolling back to a savepoint.                                                        |

## codepage\_id - ID of Code Page Used by Application

The code page identifier.

**Element identifier**

codepage\_id

**Element type**

information

*Table 60. 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 61. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |
| Connections      | event_connheader      | -              |

**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<sup>®</sup> 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.

## 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 62. 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.

## 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 63. 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.

## 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 64. 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.

## **appl\_name - Application Name**

The name of the application running at the client, as known to the database or DB2 Connect server.

**Element identifier**  
appl\_name

**Element type**  
information

*Table 65. 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 66. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |
| Activities  | event_activity        | -              |

## **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. 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\_id - Application ID**

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.

**Element identifier**  
appl\_id

**Element type**  
information

Table 67. 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 68. 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     | -              |
| Activities             | event_activitystmt        | -              |
| Activities             | event_activity            | -              |
| Activities             | event_activityvals        | -              |
| Threshold violations   | event_thresholdviolations | -              |

## 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\_appl\_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.Application instance

#### IPv4

#### Example

G91A3955.F33A.02DD18143340

#### Details

In IPv4, a TCP/IP-generated application ID is composed of three sections. The first section contains the IP address. It is represented as a 32-bit number displayed as a maximum of 8 hexadecimal characters. The second section contains the port number, which is represented as 4 hexadecimal characters. The third section contains a unique identifier for the instance of this application.

**Note:** When the hexadecimal versions of the IP address or port number begin with 0-9, they are changed to G-P respectively. For example, "0" is mapped to "G", "1" is mapped to "H", and so on.

The IP address, AC10150C.NA04.006D07064947 is interpreted as follows:

- The IP address remains AC10150C, which translates to 172.16.21.12.
- The port number is NA04. The first character is "N", which maps to "7". Therefore, the hexadecimal form of the port number is 7A04, which translates to 31236 in decimal form.

## IPv6

### Example

```
1111:2222:3333:4444:5555:6666:
7777:8888.65535.0123456789AB
```

### Details

In IPv6, a TCP/IP-generated application ID is composed of three sections. The first section contains the IP address which is a 39 byte readable address of the form a:b:c:d:e:f:g:h, where each of a-h is 4 hexadecimal digits. The second section is a readable 5-byte port number. The third section is a unique 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.

## 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.

### Element identifier

```
sequence_no
```

### Element type

```
information
```

Table 69. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 70. 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    | -              |

## 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.

### Element identifier

auth\_id

### Element type

information

Table 71. 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 72. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**Usage** You can use this element to determine who invoked the application.

## session\_auth\_id - Session Authorization ID

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.

### Element identifier

session\_auth\_id

### Element type

information

Table 73. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 74. Event Monitoring Information

| Event Type           | Logical Data Grouping | Monitor Switch |
|----------------------|-----------------------|----------------|
| Activities           | event_activity        | -              |
| Threshold violations | event_activity        | -              |

## 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.

## client\_prdid - Client Product/Version ID

The product and version that is running on the client.

### Element identifier

client\_prdid

### Element type

information

Table 75. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_id_info          | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 76. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**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\_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 77. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_id_info          | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 78. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**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.

## host\_prdid - Host Product/Version ID

The product and version that is running on the server.

### Element identifier

host\_prdid

### Element type

information

Table 79. 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 i5/OS®
  - 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)

## is\_system\_appl - Is System Application monitor element

Indicates whether the application is a system application.

### Element identifier

is\_system\_appl

### Element type

information

Table 80. 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

## 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.

**Note:** NetBIOS is no longer supported. SNA, including its APIs APPC, APPN, and CPI-C, is also no longer supported. If you use these protocols, you must recatalog your nodes and databases using a supported protocol such as TCP/IP. References to these protocols should be ignored.

### Element identifier

outbound\_appl\_id

### Element type

information

Table 81. 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

**Details**

This application ID is the displayable format of an actual SNA LUWID (Logical Unit-of-Work ID) that flows on the network when an APPC conversation is allocated. APPC-generated application IDs are made up by concatenating the network name, the LU name, and the LUWID instance number, which creates a unique label for the client/server application. The network name and LU name can each be a maximum of 8 characters. The application instance corresponds to the 12-decimal-character LUWID instance number.

## 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.

**Element identifier**

outbound\_sequence\_no

**Element type**

information

*Table 82. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dc_s_appl_info        | Basic          |

## 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.

**Element identifier**

execution\_id

**Element type**

information

*Table 83. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dc_s_appl_info        | Basic          |

*Table 84. 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.

## corr\_token - DRDA Correlation Token

The DRDA AS correlation token.

**Element identifier**  
corr\_token

**Element type**  
information

*Table 85. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Application    | appl                  | Basic          |

*Table 86. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**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.

## client\_pid - Client Process ID

The process ID of the client application that made the connection to the database.

**Element identifier**  
client\_pid

**Element type**  
information

*Table 87. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

*Table 88. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**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

The operating system on which the client application is running.

### Element identifier

client\_platform

### Element type

information

Table 89. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 90. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

**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\_protocol - Client Communication Protocol

The communication protocol that the client application is using to communicate with the server.

### Element identifier

client\_protocol

### Element type

information

Table 91. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| Application     | appl                  | Basic          |
| DCS Application | dcs_appl_info         | Basic          |

Table 92. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Connections | event_connheader      | -              |

## 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

## 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.

### Element identifier

territory\_code

### Element type

information

*Table 93. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |
| Application    | appl                  | Basic          |

*Table 94. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |
| Connections      | event_connheader      | -              |

**Usage** Territory code information is recorded in the database configuration file.

For DRDA AS connections, this element will be set to 0.

## appl\_priority - Application Agent Priority

The priority of the agents working for this application.

### Element identifier

appl\_priority

### Element type

information

*Table 95. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

*Table 96. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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 97. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 98. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**Usage** Dynamic priority is recalculated by the operating system based on usage. Static priority does not change.

## authority\_lvl - User Authorization Level

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 197.

### Element identifier

authority\_lvl

### Element type

information

Table 99. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | appl_info             | Basic          |

Table 100. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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\_SYSMaint

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\_SYSMaint\_GRP

## 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.

### Element identifier

authority\_bitmap

### Element type

information

Table 101. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | appl_info             | Basic          |

Table 102. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

## 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
```

## node\_number - Node Number

The number assigned to the node in the `db2nodes.cfg` file.

### Element identifier

`node_number`

### Element type

information

Table 103. 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          |



Table 103. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| 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 104. Event Monitoring Information

| Event Type      | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Connections     | event_connheader      | -              |
| Deadlocks       | lock                  | -              |
| Overflow Record | event_overflow        | -              |
| Database        | event_dbmemuse        | -              |
| Connection      | event_connmemuse      | -              |

**Usage** This value identifies the current node number, which can be used when monitoring multiple nodes.

## coord\_node - Coordinating Node

In a multi-node system, the node number of the node where the application connected or attached to the instance.

**Element identifier**  
coord\_node

**Element type**  
information

Table 105. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

Table 106. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**Usage** Each connected application is served by one coordinator node.

## 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 107. 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.

## 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 108. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 109. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**Usage** You may use this element to evaluate the setting of the *maxappls* configuration parameter, which is described in the *Administration Guide*.

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:

$$\text{rem\_cons\_in} + \text{local\_cons}$$

## 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 110. 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.

## 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 111. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dc_s_appl             | Unit of Work, Timestamp |

**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:

- 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 prior to 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.

## uow\_start\_time - Unit of Work Start Timestamp

The date and time that the unit of work first required database resources.

### Element identifier

uow\_start\_time

### Element type

timestamp

Table 112. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dc_s_appl             | Unit of Work, Timestamp |

**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 *uow\_stop\_time* to calculate the total elapsed time of the unit of work and with *prev\_uow\_stop\_time* 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* to calculate the elapsed time for the SQL Reference's definition of a unit of work.

## uow\_stop\_time - Unit of Work Stop Timestamp

The date and time that the most recent unit of work completed, which occurs when database changes are committed or rolled back.

### Element identifier

uow\_stop\_time

### Element type

timestamp

Table 113. 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** You may use this element with *prev\_uow\_stop\_time* to calculate the total elapsed time between COMMIT/ROLLBACK points, and with *uow\_start\_time* 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 *uow\_start\_time*). this element will be a valid, non-zero timestamp
- When the application is currently executing a unit of work, this element will contain zeros
- When the application first connects to the database, this element is set to *conn\_complete\_time*.

As a new unit of work is started, the contents of this element are moved to *prev\_uow\_stop\_time*.

## 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 114. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch          |
|-----------------|-----------------------|-------------------------|
| Application     | appl                  | Unit of Work, Timestamp |
| DCS Application | dc_s_appl             | Unit of Work, Timestamp |

**Usage** Use this element as an indicator of the time it takes for units of work to complete.

## 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 115. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl                  | Unit of Work   |
| DCS Application | dc_s_appl             | Basic          |

Table 116. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Transactions | event_xact            | -              |

**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\_status - Unit of Work Status

The status of the unit of work.

### Element identifier

uow\_status

### Element type

information

Table 117. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Transactions | event_xact            | -              |

**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.

## 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 118. 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.

## DB2 agent information monitor elements

The following database system monitor elements provide information about agents.

### agent\_pid - Engine dispatchable unit (EDU) 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.

**Element identifier**  
agent\_pid

**Element type**  
information

*Table 119. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | agent                 | Statement      |

**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.

### coord\_agent\_pid - Coordinator Agent 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.

**Element identifier**  
coord\_agent\_pid

**Element type**  
information

Table 120. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |

**Usage** You can use this element to link database system monitor information to other sources of diagnostic information, such as system traces.

---

## Database manager configuration monitor elements

The following database manager monitor elements can be used to monitor the progress/activity of some DB2 functions. For example, one type of DB2 function that supports progress monitoring is the backup utility.

Some DB2 functions consist of a single step of processing which can be described in the monitor stream under a single `progress_info` logical grouping. More complicated DB2 functions consist of multiple steps of execution. For example, the LOAD utility consists of three core phases: LOAD, BUILD and DELETE. A multi-step function, will be described by a `progress_info_list` logical group which will contain a `progress_info` logical grouping to describe each unique phase of the utility.

Some functions may report different elements than others. For example, some DB2 utilities may not be able to quantify the total amount of work they will perform and thus will not specify a `progress_info_total_work_units` element

## Agents and connections monitor elements

An agent is a process or thread that carries out the requests made by a client application. Each connected application is served by exactly 1 *coordinator agent* and possibly, a set of subordinator agents or *subagents*. Subagents are used for parallel SQL processing in partitioned databases and on SMP machines. Agents are classified as follows:

### Coordinator agent

This is the initial agent to which a local or remote application connects. There is one coordinator agent dedicated to each database connection or instance attachment. The maximum number of coordinating agents per partition is controlled by the `max_coordagents` configuration parameter.

### Subagent

In partitioned databases, additional agents can be enlisted by the coordinator agent to speed up SQL processing. Subagents are selected from the agent pool and are returned there when their work is done. The size of the agent pool is controlled by the `num_poolagents` configuration parameter.

### Associated agent

A coordinator or subagent that is doing work for an application is associated with that application. After it is finished an application's work, it goes into the agent pool as an associated agent. If the application attempts to do more work, DB2 will search the agent pool for an agent

already associated with the application and assign the work to it. If none is found, DB2 will attempt to get an agent to satisfy the request by:

1. Choosing an idle agent that is not associated with an application.
2. Finding an agent that is associated with another application. If an idle agent cannot be found on the current application, DB2 will try to take an idle agent associated with another application. This is referred to as a *stolen agent*.
3. Creating an agent, if an idle agent is not available.

### Primed agent

A gateway agent in the DRDA connections pool that is connected to a DRDA database in anticipation of work on the remote database.

The initial number of agents that are created in the agent pool at DB2START is determined by the **num\_initagents** configuration parameter.

Assuming no idle agents, each connection creates a new agent, unless **max\_coordagents** has been reached.

The following elements provide agent and connection information.

### 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.

#### Element identifier

rem\_cons\_in

#### Element type

gauge

Table 121. 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.

#### Element identifier

rem\_cons\_in\_exec



**Element type**  
gauge

Table 122. 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`.

## local\_cons - Local Connections

The number of local applications that are currently connected to a database within the database manager instance being monitored.

**Element identifier**  
local\_cons

**Element type**  
gauge

Table 123. 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.

### Element identifier

local\_cons\_in\_exec

### Element type

gauge

Table 124. 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**.

## con\_local\_dbases - Local Databases with Current Connects

The number of local databases that have applications connected.

### Element identifier

con\_local\_dbases

### Element type

gauge

Table 125. 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

## total\_cons - Connects Since Database Activation

Indicates the number of connections to the database since the first connect, activate, or last reset (coordinator agents).

### Element identifier

total\_cons

### Element type

counter

Table 126. 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 127. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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 overhead 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.

## appls\_cur\_cons - Applications Connected Currently

Indicates the number of applications that are currently connected to the database.

### Element identifier

appls\_cur\_cons

### Element type

gauge

Table 128. 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.

### Element identifier

appls\_in\_db2

### Element type

gauge

Table 129. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

## agents\_registered - Agents Registered

The number of agents registered in the database manager instance that is being monitored (coordinator agents and subagents).

### Element identifier

agents\_registered

### Element type

gauge

Table 130. 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\_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.

### Element identifier

agents\_waiting\_on\_token

### Element type

gauge

Table 131. 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\_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).

#### **Element identifier**

agents\_registered\_top

#### **Element type**

watermark

*Table 132. 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\_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.

#### **Element identifier**

agents\_waiting\_top

#### **Element type**

watermark

*Table 133. 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.

### 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".

#### Element identifier

idle\_agents

#### Element type

gauge

Table 134. 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.

### agents\_from\_pool - Agents Assigned From Pool

The number of agents assigned from the agent pool.

#### Element identifier

agents\_from\_pool

#### Element type

counter

Table 135. 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\_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 DB2 start up (*num\_initagents*).

**Element identifier**

agents\_created\_empty\_pool

**Element type**

counter

*Table 136. 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 on using this element.

### **coord\_agents\_top - Maximum Number of Coordinating Agents**

The maximum number of coordinating agents working at one time.

**Element identifier**

coord\_agents\_top

**Element type**

watermark

*Table 137. 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.

### **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.

**Element identifier**  
agents\_stolen

**Element type**  
counter

*Table 138. 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.

### **associated\_agents\_top - Maximum Number of Associated Agents**

The maximum number of subagents associated with this application.

**Element identifier**  
associated\_agents\_top

**Element type**  
watermark

*Table 139. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

### **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.

**Element identifier**  
comm\_private\_mem

**Element type**  
gauge

*Table 140. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

### **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 141. 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.

### **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 142. 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.

### **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.

#### **Element identifier**

max\_agent\_overflows

#### **Element type**

gauge

Table 143. 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.

### **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.

**Element identifier**  
num\_gw\_conn\_switches

**Element type**  
gauge

Table 144. 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.

## Memory pool monitor elements

Database-wide memory pools are reported in database snapshots, and instance-wide memory pools are reported in database manager snapshots.

The following elements provide information about the memory pools.

### pool\_id - Memory Pool Identifier

The type of memory pool.

**Element identifier**  
pool\_id

**Element type**  
Information

Table 145. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 146. 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\_max\_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\_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 147. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 148. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_dbmemuse        | -              |
| Connection | event_connmemuse      | -              |

**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\_cur\_size - Current Size of Memory Pool

The current size of a memory pool.

### Element identifier

`pool_cur_size`

### Element type

Information

Table 149. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 150. 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_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\_config\_size - Configured Size of Memory Pool

The internally configured size of a memory pool in DB2 database system.

### Element identifier

`pool_config_size`

### Element type

Information

Table 151. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 152. 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\_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\_watermark - Memory Pool Watermark

The largest size of a memory pool since its creation.

#### Element identifier

pool\_watermark

#### Element type

Information

Table 153. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | memory_pool           | Basic          |
| Database         | memory_pool           | Basic          |
| Application      | memory_pool           | Basic          |

Table 154. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_dbmemuse        | -              |
| Connection | event_connmemuse      | -              |

**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 instance, 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.

## Sort monitor elements

The following elements provide information about the database manager sort work performed.

## 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.

### Element identifier

sort\_heap\_allocated

### Element type

gauge

Table 155. 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 *sheaphres* configuration parameter. If the element value is greater than or equal to *sheaphres*, it means that the sorts are not getting the full sort heap as defined by the *sortheap* parameter.

## post\_threshold\_sorts - Post Threshold Sorts

The number of sorts that have requested heaps after the sort heap threshold has been exceeded.

### Element identifier

post\_threshold\_sorts

### Element type

counter

Table 156. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Sort           |

For snapshot monitoring, this counter can be reset.

**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.

### **post\_shrthreshold\_sorts - Post shared threshold sorts**

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. 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.

#### **Element identifier**

post\_shrthreshold\_sorts

#### **Element type**

counter

*Table 157. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Sort           |

For snapshot monitoring, this counter can be reset.

*Table 158. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

### **pipedsortsrequested - Piped Sorts Requested**

The number of piped sorts that have been requested.

#### **Element identifier**

pipedsortsrequested

#### **Element type**

counter

*Table 159. 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 be accepted if the sort heap threshold will be exceeded when the sort heap is allocated for the sort. See *pipedsorts\_accepted* for more information if you are experiencing piped sort rejections.

The SQL EXPLAIN output will show whether the optimizer requests a piped sort. For more information on piped and non-piped sorts see the *Administration Guide*.

### **pipedsorts\_accepted - Piped Sorts Accepted**

The number of piped sorts that have been accepted.

**Element identifier**

pipedsorts\_accepted

**Element type**

counter

*Table 160. 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.

See the *Administration Guide* for more information on sorts.

### **totalsorts - Total Sorts**

The total number of sorts that have been executed.

**Element identifier**

totalsorts



**Element type**  
counter

Table 161. 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 162. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch  |
|------------|-----------------------|-----------------|
| Database   | event_db              | -               |
| Connection | event_conn            | -               |
| Statements | event_stmt            | -               |
| Activities | event_activity        | Statement, Sort |

## 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. See the *Administration Guide* for more information.

## **total\_sort\_time - Total Sort Time**

The total elapsed time (in milliseconds) for all sorts that have been executed.

**Element identifier**  
total\_sort\_time

**Element type**  
counter

Table 163. 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 164. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch  |
|------------|-----------------------|-----------------|
| Database   | event_db              | -               |
| Connection | event_conn            | -               |
| Statements | event_stmt            | -               |
| 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:

$$\text{total\_sort\_time} / \text{total\_sorts}$$

provides information about the average elapsed time for each sort.

## sort\_overflows - Sort Overflows

The total number of sorts that ran out of sort heap and may have required disk space for temporary storage.

### Element identifier

sort\_overflows

### Element type

counter

Table 165. 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 166. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch  |
|------------|-----------------------|-----------------|
| Database   | event_db              | -               |
| Connection | event_conn            | -               |
| Statements | event_stmt            | -               |
| Activities | event_activity        | Statement, Sort |

## 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 overhead will be incurred 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.

## active\_sorts - Active Sorts

The number of sorts in the database that currently have a sort heap allocated.

### Element identifier

active\_sorts

### Element type

gauge

Table 167. 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. (See the *Administration Guide* for more details.)

This value includes heaps for sorts of temporary tables that were created during relational operations.

## sort\_heap\_top - Sort private heap high watermark

The private sort memory high watermark, in 4 KB pages, across the database manager.

### Element identifier

sort\_heap\_top

**Element type**  
watermark

*Table 168. 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\_shrheap\_allocated - Sort Share Heap Currently Allocated**

Total amount of shared sort memory allocated in the database.

**Element identifier**  
sort\_shrheap\_allocated

**Element type**  
information

*Table 169. 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.

**Element identifier**  
sort\_shrheap\_top

**Element type**  
watermark

*Table 170. 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.

## Hash join monitor elements

Hash join is an additional option for the optimizer. A hash join will first compare *hash codes* before comparing predicates for tables involved in a join. In a hash join, one table (selected by the optimizer) is scanned and rows are copied into memory buffers drawn from the sort heap allocation. The memory buffers are divided into partitions based on a hash code computed from the columns of the join predicates. Rows of the other table involved in the join are matched to rows from the first table by comparing the hash code. If the hash codes match, the actual join predicate columns are compared.

### total\_hash\_joins - Total Hash Joins

The total number of hash joins executed.

#### Element identifier

total\_hash\_joins

#### Element type

counter

Table 171. 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 172. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### 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 173. 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\_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.

**Element identifier**  
post\_shrthreshold\_hash\_joins

**Element type**  
counter

Table 174. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

For snapshot monitoring, this counter can be reset.

Table 175. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

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.

### active\_hash\_joins - Active hash joins

The total number of hash joins that are currently running and consuming memory.

**Element identifier**  
active\_hash\_joins

**Element type**  
counter

Table 176. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

## 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.

### Element identifier

total\_hash\_loops

### Element type

counter

Table 177. 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 178. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

## 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 179. 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 180. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 181. 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 182. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

## On-Line Analytical Processing (OLAP) monitor elements

The following monitor elements provide information about OLAP functions.

Use these monitor elements, along with similar sort and hash join monitor elements, to help diagnose sort heap problems and tune sort heap usage.

**Note:** The SQL compiler often combines the execution of several compatible OLAP functions into one runtime operation. As a result, the following counts may appear smaller than expected. For example, an SQL statement could contain references to four OLAP functions which, when executed, might result in the **total\_olap\_funcs** monitor element value only being incremented once.

### total\_olap\_funcs - Total OLAP Functions monitor element

The total number of OLAP functions executed.

### Element identifier

total\_olap\_funcs

### Element type

counter

Table 183. 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 184. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

## 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.

### **olap\_func\_overflows - OLAP Function Overflows monitor element**

The number of times that OLAP function data exceeded the available sort heap space.

#### Element identifier

`olap_func_overflows`

#### Element type

counter

Table 185. 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 186. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

## 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.

### **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.

#### Element identifier

`post_threshold_olap_funcs`

**Element type**  
counter

*Table 187. 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 utilize 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`).

### **active\_olap\_funcs - Active OLAP Functions monitor element**

The total number of OLAP functions that are currently running and consuming sort heap memory.

**Element identifier**  
`active_olap_funcs`

**Element type**  
counter

*Table 188. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

For snapshot monitoring, this counter can be reset.

## Fast communications manager (FCM) monitor elements

The following database system monitor elements provide information about the Fast Communication Manager (FCM).

### **buff\_free - FCM Buffers Currently Free**

This element indicates the number of FCM buffers currently free.

**Element identifier**  
`buff_free`

**Element type**  
gauge

Table 189. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

**Usage** Use the number of FCM buffers currently free in conjunction with the *fcm\_num\_buffers* configuration parameter to determine the current FCM buffer pool utilization. You can use this information to tune *fcm\_num\_buffers*.

### buff\_free\_bottom - Minimum FCM Buffers Free

The lowest number of free FCM buffers reached during processing.

**Element identifier**

buff\_free\_bottom

**Element type**

watermark

Table 190. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

**Usage** Use this element in conjunction with the *fcm\_num\_buffers* configuration parameter to determine the maximum FCM buffer pool utilization. If buff\_free\_bottom is low, you should increase *fcm\_num\_buffers* to ensure that operations do not run out of FCM buffers. If buff\_free\_bottom is high, you can decrease *fcm\_num\_buffers* to conserve system resources.

### connection\_status - Connection Status

This element indicates the status of the communication connection status between the node issuing the GET SNAPSHOT command and other nodes listed in the *db2nodes.cfg* file.

**Element identifier**

connection\_status

**Element type**

information

Table 191. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

**Usage** The connection values are :

**SQLM\_FCM\_CONNECT\_INACTIVE**

No current connection

**SQLM\_FCM\_CONNECT\_ACTIVE**

Connection is active

**SQLM\_FCM\_CONNECT\_CONGESTED**

Connection is congested

Two nodes can be active, but the communication connection between them will remain inactive, unless there is some communication between those nodes.

### **total\_buffers\_sent - Total FCM Buffers Sent**

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* (see the *db2nodes.cfg* file).

**Element identifier**

total\_buffers\_sent

**Element type**

counter

*Table 192. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

**Usage** You can use this element to measure the level of traffic between the current node and the remote node. If the total number of FCM buffers sent to this node is high, you may want to redistribute the database, or move tables to reduce the inter-node traffic.

### **total\_buffers\_rcvd - Total FCM Buffers Received**

The total number of FCM buffers received by the node issuing the GET SNAPSHOT command from the node identified by the *node\_number* (see the *db2nodes.cfg* file).

**Element identifier**

total\_buffers\_rcvd

**Element type**

counter

*Table 193. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm_node              | Basic          |

**Usage** You can use this element to measure the level of traffic between the current node and the remote node. If the total number of FCM buffers received from this node is high, you may want to redistribute the database, or move tables to reduce the inter-node traffic.

### **ch\_free - Channels Currently Free**

This element indicates the number of inter-node communication channels that are currently free.

**Element identifier**

ch\_free

**Element type**

gauge

*Table 194. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fcm                   | Basic          |

**Usage** Use the number of communication channels currently free in conjunction

with the *fcnum\_channels* configuration parameter to determine the current connection entry utilization. You can use this information to tune *fcnum\_channels*.

### **ch\_free\_bottom - Minimum Channels Free**

The lowest number of free inter-node communication channels reached during processing.

**Element identifier**

ch\_free\_bottom

**Element type**

watermark

*Table 195. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | fc                    | Basic          |

**Usage** Use this element in conjunction with the *fcnum\_channels* configuration parameter to determine the maximum connection entry utilization.

---

## **Database configuration monitor elements**

The following elements provide information particularly helpful for database performance tuning.

### **Buffer pool activity monitor elements**

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.

The buffer pool hit ratio can be calculated as follows:

$$1 - \left( \frac{\text{pool\_data\_p\_reads} + \text{pool\_xda\_p\_reads} + \text{pool\_index\_p\_reads} + \text{pool\_temp\_data\_p\_reads} + \text{pool\_temp\_xda\_p\_reads} + \text{pool\_temp\_index\_p\_reads}}{\text{pool\_data\_l\_reads} + \text{pool\_xda\_l\_reads} + \text{pool\_index\_l\_reads} + \text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads} + \text{pool\_temp\_index\_l\_reads}} \right) * 100\%$$

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 desired result. This can be achieved using two methods:

1. Split the data and indices 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:

$$(1 - ((\text{pool\_index\_p\_reads}) / (\text{pool\_index\_l\_reads}))) * 100\%$$

The first method is often more effective, but because it requires indices 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 desired 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.

The following elements provide information about buffer pool activity.

### **bp\_id - Buffer pool identifier monitor element**

This element contains the buffer pool identifier for the buffer pool that is being monitored.

**Element identifier**

bp\_id

**Element type**

information

*Table 196. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Basic          |

### **pool\_data\_l\_reads - Buffer Pool Data Logical Reads**

Indicates the number of data pages which have been requested from the buffer pool (logical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

**Element identifier**

pool\_data\_l\_reads

**Element type**

counter

*Table 197. 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 198. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

### **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.

In conjunction with **pool\_data\_p\_reads**, you can calculate the data page hit ratio for the buffer pool using the following formula:

$$1 - ((\text{pool\_data\_p\_reads} - \text{pool\_async\_data\_reads}) / \text{pool\_data\_l\_reads})$$

For information about determining the overall buffer pool hit ratio, see “Buffer pool activity monitor elements” on page 235.

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. In such case, you would focus your attention on smaller, frequently accessed tables, and on the indices. Perhaps, assigning them to an individual buffer pools, for which you can aim for higher hit ratios.

### **pool\_temp\_data\_l\_reads - Buffer Pool Temporary Data Logical Reads**

Indicates the number of data pages which have been requested from the buffer pool (logical) for temporary table spaces.

**Element identifier**

pool\_temp\_data\_l\_reads

**Element type**

counter

*Table 199. 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 200. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespaces     | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |



## 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\_p\_reads** element, a calculation for the data page hit ratio for buffer pools located in temporary table spaces can be made using the following formula:

$$1 - (\text{pool\_temp\_data\_p\_reads} / \text{pool\_temp\_data\_l\_reads})$$

For information about determining the overall buffer pool hit ratio, see “Buffer pool activity monitor elements” on page 235.

### **pool\_data\_p\_reads - Buffer Pool Data Physical Reads**

Indicates the number of data pages read in from the table space containers (physical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

#### **Element identifier**

pool\_data\_p\_reads

#### **Element type**

counter

*Table 201. 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 202. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespaces     | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

See “pool\_data\_l\_reads - Buffer Pool Data Logical Reads” on page 237 and “pool\_async\_data\_reads - Buffer pool asynchronous data reads monitor element” on page 252 for information about how to use this element.

## pool\_temp\_data\_p\_reads - Buffer Pool Temporary Data Physical Reads

Indicates the number of data pages read in from the table space containers (physical) for temporary table spaces.

### Element identifier

pool\_temp\_data\_p\_reads

### Element type

counter

Table 203. 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 204. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

For information about how to use this element, see “pool\_temp\_data\_l\_reads - Buffer Pool Temporary Data Logical Reads” on page 238.

## pool\_data\_writes - Buffer Pool Data Writes

Indicates the number of times a buffer pool data page was physically written to disk.

### Element identifier

pool\_data\_writes

### Element type

counter

Table 205. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |

Table 205. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 206. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespaces     | -              |
| Connection  | event_conn            | -              |

**Usage** If a buffer pool data page is written to disk for a high percentage of the pool\_data\_p\_reads, 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. These asynchronous page writes are included in the value of this element in addition to synchronous page writes (see pool\_async\_data\_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 either;

- 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.

See the *Administration Guide* for more information about buffer pool size.

## pool\_index\_l\_reads - Buffer Pool Index Logical Reads

Indicates the number of index pages which have been requested from the buffer pool (logical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

### Element identifier

pool\_index\_l\_reads

### Element type

counter

Table 207. 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 208. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## 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**, you can calculate the index page hit ratio for the buffer pool using one of the following:

$$1 - ((\text{pool\_index\_p\_reads} - \text{pool\_async\_index\_reads}) / \text{pool\_index\_l\_reads})$$

To calculate the overall buffer pool hit ratio, see “pool\_data\_l\_reads - Buffer Pool Data Logical Reads” on page 237.

If the hit ratio is low, increasing the number of buffer pool pages may improve performance. See the *Administration Guide* for more information about buffer pool size.

## pool\_temp\_index\_l\_reads - Buffer Pool Temporary Index Logical Reads

Indicates the number of index pages which have been requested from the buffer pool (logical) for temporary table spaces.

### Element identifier

pool\_temp\_index\_l\_reads

### Element type

counter

Table 209. 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 210. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

For information about how to use this element, see “pool\_temp\_data\_l\_reads - Buffer Pool Temporary Data Logical Reads” on page 238.

## pool\_index\_p\_reads - Buffer Pool Index Physical Reads

Indicates the number of index pages read in from the table space containers (physical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

### Element identifier

pool\_index\_p\_reads

### Element type

counter

Table 211. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

Table 211. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| 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 212. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

See “pool\_index\_l\_reads - Buffer Pool Index Logical Reads” on page 242 for information about how to use this element.

### pool\_temp\_index\_p\_reads - Buffer Pool Temporary Index Physical Reads

Indicates the number of index pages read in from the table space containers (physical) for temporary table spaces.

#### Element identifier

pool\_temp\_index\_p\_reads

#### Element type

counter

Table 213. 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 214. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

Table 214. Event Monitoring Information (continued)

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

For information about how to use this element, see “pool\_temp\_data\_l\_reads - Buffer Pool Temporary Data Logical Reads” on page 238.

### pool\_index\_writes - Buffer Pool Index Writes

Indicates the number of times a buffer pool index page was physically written to disk.

#### Element identifier

pool\_index\_writes

#### Element type

counter

Table 215. 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 216. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |

**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 `pool_async_index_writes`).

If a buffer pool index page is written to disk for a high percentage of the `pool_index_p_reads`, 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 either:

- 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.

See the *Administration Guide* for more information about buffer pool size.

### **pool\_xda\_l\_reads - Buffer Pool XDA Data Logical Reads**

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.

#### **Element identifier**

`pool_xda_l_reads`

#### **Element type**

counter

*Table 217. 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 218. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |
| Statement   | event_stmt            | -              |



Table 218. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch         |
|------------|-----------------------|------------------------|
| Activities | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

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\_xda\_l\_reads** monitor element in conjunction with **pool\_xda\_p\_reads**, **pool\_data\_l\_reads**, and **pool\_data\_p\_reads** to calculate the data page hit ratio for the buffer pool by using the following formula:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_xda\_p\_reads}) / (\text{pool\_data\_l\_reads} + \text{pool\_xda\_l\_reads}))$$

For information about determining the overall buffer pool hit ratio, see “Buffer pool activity monitor elements” on page 235.

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. In such case, you would focus your attention on smaller, frequently accessed tables, and on the indices. Perhaps, assigning them to an individual buffer pools, for which you can aim for higher hit ratios.

## pool\_temp\_xda\_l\_reads - Buffer Pool Temporary XDA Data Logical Reads

Indicates the number of pages for XML storage object (XDA) data which have been requested from the buffer pool (logical) for temporary table spaces.

### Element identifier

pool\_temp\_xda\_l\_reads

### Element type

counter

Table 219. 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    |

Table 219. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch         |
|----------------|-----------------------|------------------------|
| Dynamic SQL    | dynsql                | Buffer Pool, Statement |

For snapshot monitoring, this counter can be reset.

Table 220. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

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** to calculate the data page hit ratio for buffer pools located in temporary table spaces by using the following formula:

$$1 - ((\text{pool\_temp\_data\_p\_reads} + \text{pool\_temp\_xda\_p\_reads}) / (\text{pool\_temp\_data\_l\_reads} + \text{pool\_temp\_xda\_l\_reads}))$$

For information about determining the overall buffer pool hit ratio, see “Buffer pool activity monitor elements” on page 235.

### pool\_xda\_p\_reads - Buffer Pool XDA Data Physical Reads

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.

#### Element identifier

pool\_xda\_p\_reads

#### Element type

counter

Table 221. 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 222. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

For information about how to use this element, see “pool\_xda\_l\_reads - Buffer Pool XDA Data Logical Reads” on page 246 and “pool\_async\_xda\_reads - Buffer Pool Asynchronous XDA Data Reads” on page 255.

### **pool\_temp\_xda\_p\_reads - Buffer Pool Temporary XDA Data Physical Reads**

Indicates the number of pages for XML storage object (XDA) data read in from the table space containers (physical) for temporary table spaces.

#### Element identifier

pool\_temp\_xda\_p\_reads

#### Element type

counter

Table 223. 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 224. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch         |
|-------------|-----------------------|------------------------|
| Database    | event_db              | -                      |
| Tablespaces | event_tablespace      | -                      |
| Connection  | event_conn            | -                      |
| Statement   | event_stmt            | -                      |
| Activities  | event_activity        | Buffer Pool, Statement |

## Usage

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

For information about how to use this element, see “pool\_temp\_xda\_l\_reads - Buffer Pool Temporary XDA Data Logical Reads” on page 247.

### pool\_xda\_writes - Buffer Pool XDA Data Writes

Indicates the number of times a buffer pool data page for an XML storage object (XDA) was physically written to disk.

#### Element identifier

pool\_xda\_writes

#### Element type

counter

Table 225. 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 226. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |

**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).

See pool\_xda\_l\_reads and pool\_xda\_p\_reads for more information about how to use this element.

See the *Administration Guide* for more information about buffer pool size.

### pool\_read\_time - Total Buffer Pool Physical Read Time

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.

#### Element identifier

pool\_read\_time

#### Element type

counter

Table 227. 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 228. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |

**Usage** You can use this element with *pool\_data\_p\_reads* and *pool\_index\_p\_reads* 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 *pool\_async\_read\_time*.

### **pool\_write\_time - Total Buffer Pool Physical Write Time**

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.

#### **Element identifier**

pool\_write\_time

#### **Element type**

counter

Table 229. 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 230. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |

**Usage** You can use this element with *buffer\_pool\_data\_writes* and *pool\_index\_writes* 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 *pool\_async\_write\_time*.

### **files\_closed - Database Files Closed**

The total number of database files closed.

**Element identifier**

files\_closed

**Element type**

counter

*Table 231. 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 232. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**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 (see the *Administration Guide* for more information).

### **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.

**Element identifier**

pool\_async\_data\_reads

**Element type**

counter

*Table 233. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |

Table 233. 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 234. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

## 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:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads}) - (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads})) / (\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads})$$

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. For information about these prefetchers, see “Prefetching data into the buffer pool” in *Tuning Database Performance*.

## pool\_async\_data\_writes - Buffer Pool Asynchronous Data Writes

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.

### Element identifier

pool\_async\_data\_writes

### Element type

counter

Table 235. 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 236. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**Usage** You can use this element with `buffer_pool_data_writes` 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_io cleaners` configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.

### **pool\_async\_index\_writes - Buffer Pool Asynchronous Index Writes**

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.

#### **Element identifier**

`pool_async_index_writes`

#### **Element type**

counter

*Table 237. 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 238. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**Usage** You can use this element with `pool_index_writes` 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_io cleaners` configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.



## pool\_async\_index\_reads - Buffer Pool Asynchronous Index Reads

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.

### Element identifier

pool\_async\_index\_reads

### Element type

counter

Table 239. 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 240. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**Usage** You can use this element with pool\_index\_p\_reads 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:

$$1 - ((\text{pool\_data\_p\_reads} + \text{pool\_index\_p\_reads}) - (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads}))$$

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 (see the *Administration Guide*).

Asynchronous reads are performed by database manager prefetchers. For information about these prefetchers, see the *Administration Guide*.

## pool\_async\_xda\_reads - Buffer Pool Asynchronous XDA Data Reads

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.

### Element identifier

pool\_async\_xda\_reads

### Element type

counter

Table 241. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

Table 241. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 242. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespaces     | -              |

**Usage** You can use this element with `pool_xda_p_reads` 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 (see the *Administration Guide*).

Asynchronous reads are performed by database manager prefetchers. For information about these prefetchers, see the *Administration Guide*.

### **pool\_async\_xda\_writes - Buffer Pool Asynchronous XDA Data Writes**

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.

#### **Element identifier**

`pool_async_xda_writes`

#### **Element type**

counter

Table 243. 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 244. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespaces     | -              |

**Usage** You can use this element with `pool_xda_writes` 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_io cleaners` configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.

### **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 245. 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 246. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**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**

The total elapsed time spent writing data or index pages from the buffer pool to disk by database manager page cleaners.

**Element identifier**

pool\_async\_write\_time

**Element type**

counter

*Table 247. 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 248. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**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 read 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\_data\_read\_reqs - Buffer Pool Asynchronous Read Requests**

The number of asynchronous read requests.

**Element identifier**

pool\_async\_data\_read\_reqs

**Element type**

counter

*Table 249. 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 250. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

Table 250. Event Monitoring Information (continued)

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Tablespaces | event_tablespace      | -              |

**Usage** To calculate the average number of data pages read per asynchronous request, use the following formula:

$$\text{pool\_async\_data\_reads} / \text{pool\_async\_data\_read\_reqs}$$

This average can help you determine the amount of asynchronous I/O done in each interaction with the prefetcher.

### **pool\_async\_index\_read\_reqs - Buffer Pool Asynchronous Index Read Requests**

The number of asynchronous read requests for index pages.

**Element identifier**

pool\_async\_index\_read\_reqs

**Element type**

counter

Table 251. 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 252. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**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\_xda\_read\_reqs - Buffer Pool Asynchronous XDA Read Requests**

The number of asynchronous read requests for XML storage object (XDA) data.

**Element identifier**

pool\_async\_xda\_read\_reqs

**Element type**

counter

Table 253. 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 254. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |

**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\_lsn\_gap\_clns - Buffer Pool Log Space Cleaners Triggered**

The number of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.

#### **Element identifier**

pool\_lsn\_gap\_clns

#### **Element type**

counter

Table 255. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 256. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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. See the *Administration Guide* for more information.

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\_drty\_pg\_steal\_clns - Buffer Pool Victim Page Cleaners Triggered**

The number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

#### **Element identifier**

pool\_drty\_pg\_steal\_clns

#### **Element type**

counter

*Table 257. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

*Table 258. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**Usage** Using the following formula, you may calculate what percentage of all cleaner invocations are represented by this element:

$$\frac{\text{pool\_drty\_pg\_steal\_clns}}{\text{pool\_drty\_pg\_steal\_clns} + \text{pool\_drty\_pg\_thrsh\_clns} + \text{pool\_lsn\_gap\_clns}}$$

If this ratio is low, it may indicate that you have defined too many page cleaners. If your *chnngpgs\_thresh* 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 too few page cleaners defined. Too few page cleaners will increase recovery time after failures (see the *Administration Guide*).

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_dirty_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, please 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\_no\_victim\_buffer - Buffer Pool No Victim Buffers**

Number of times an agent did not have a preselected victim buffer available.

#### **Element identifier**

`pool_no_victim_buffer`

#### **Element type**

counter

*Table 259. 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 260. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Tablespace | event_tablespace      | -              |

**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\_drty\_pg\_thrsh\_clns - Buffer Pool Threshold Cleaners Triggered

The number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

### Element identifier

pool\_drty\_pg\_thrsh\_clns

### Element type

counter

Table 261. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 262. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**Usage** The threshold is set by the *chngpgs\_thresh* 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 this value is set too low, pages might be written out too early, requiring them to be read back in. If set too high, then too many pages may accumulate, requiring users to write out pages synchronously. See the *Administration Guide* for more information.

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.

## bp\_name - Buffer Pool Name

The name of the buffer pool.

### Element identifier

bp\_name

### Element type

information

Table 263. 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.

### prefetch\_wait\_time - Time Waited for Prefetch

The time an application spent waiting for an I/O server (prefetcher) to finish loading pages into the buffer pool.

**Element identifier**

prefetch\_wait\_time

**Element type**

counter

Table 264. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

Table 265. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**Usage** This element can be used to experiment with changing the number of I/O servers, and I/O server sizes.

### unread\_prefetch\_pages - Unread Prefetch Pages

Indicates the number of pages that the prefetcher read in that were never used.

**Element identifier**

unread\_prefetch\_pages

**Element type**

counter

Table 266. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |

Table 266. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Buffer Pool    |
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 267. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tablespaces | event_tablespace      | -              |
| Connection  | event_conn            | -              |

**Usage** If this number is high, prefetchers are causing unnecessary I/O by reading pages into the buffer pool that will not be used. See the *Administration Guide* for more information about prefetching.

### vectored\_ios - Number of Vectored IO Requests

The number of vectored I/O requests. More specifically, the number of times DB2 performs sequential prefetching of pages into the page area of the buffer pool.

**Element identifier**

vectored\_ios

**Element type**

gauge

Table 268. 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 sequential prefetching.

### pages\_from\_vectored\_ios - Total Number of Pages Read by Vectored IO

The total number of pages read by vectored I/O into the page area of the buffer pool.

**Element identifier**

pages\_from\_vectored\_ios

**Element type**

gauge

Table 269. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |

## block\_ios - Number of Block IO Requests

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.

### Element identifier

block\_ios

### Element type

counter

Table 270. 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 Number of Vectored IO Requests monitor element), consider changing the block size. This state can be an indication of the following:

- 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.

## pages\_from\_block\_ios - Total Number of Pages Read by Block IO

The total number of pages read by block I/O into the block area of the buffer pool.

### Element identifier

pages\_from\_block\_ios

### Element type

counter

Table 271. 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 will contain the total number of pages read by block I/O. Otherwise, this element will return 0.

*pages\_from\_block\_ios* divided by the *block\_ios* element gives an average number of pages sequentially prefetched per block-based I/O. If *pages\_from\_block\_ios* divided by *block\_ios* is much less than the BLOCKSIZE you have defined for the block-based buffer pool, 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 bufferpool.

### Dynamic buffer pool monitor elements

The following monitor elements provide information about dynamic buffer pools.

#### bp\_cur\_buffsz - Current Size of Buffer Pool:

Current buffer pool size.

**Element identifier**

bp\_cur\_buffsz

**Element type**

gauge

Table 272. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

#### 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.

**Element identifier**

bp\_new\_buffsz

**Element type**

information

Table 273. 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.

**Element identifier**

bp\_pages\_left\_to\_remove

**Element type**  
gauge

*Table 274. 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.

**Element identifier**  
bp\_tbsp\_use\_count

**Element type**  
gauge

*Table 275. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool_nodeinfo   | Buffer Pool    |

## **Non-buffered I/O activity monitor elements**

The following elements provide information about I/O activity that does not use the buffer pool.

### **direct\_reads - Direct Reads From Database**

The number of read operations that do not use the buffer pool.

**Element identifier**  
direct\_reads

**Element type**  
counter

*Table 276. 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 277. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Connection  | event_conn            | -              |
| Tablespaces | event_tablespace      | -              |

**Usage** Use the following formula to calculate the average number of sectors that are read by a direct read:

`direct_reads / 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\_writes - Direct Writes to Database**

The number of write operations that do not use the buffer pool.

#### **Element identifier**

`direct_writes`

#### **Element type**

counter

*Table 278. 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 279. Event Monitoring Information*

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Connection  | event_conn            | -              |
| Tablespaces | event_tablespace      | -              |

**Usage** Use the following formula to calculate the average number of sectors that are written by a direct write.

`direct_writes / 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)

## direct\_read\_reqs - Direct Read Requests

The number of requests to perform a direct read of one or more sectors of data.

### Element identifier

direct\_read\_reqs

### Element type

counter

Table 280. 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 281. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Connection  | event_conn            | -              |
| Tablespaces | event_tablespace      | -              |

**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\_write\_reqs - Direct Write Requests

The number of requests to perform a direct write of one or more sectors of data.

### Element identifier

direct\_write\_reqs

### Element type

counter

Table 282. 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 283. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |



Table 283. Event Monitoring Information (continued)

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Tablespaces | event_tablespace      | -              |

**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\_read\_time - Direct Read Time

The elapsed time (in milliseconds) required to perform the direct reads.

**Element identifier**

direct\_read\_time

**Element type**

counter

Table 284. 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 285. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Connection  | event_conn            | -              |
| Tablespaces | event_tablespace      | -              |

**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\_write\_time - Direct Write Time

The elapsed time (in milliseconds) required to perform the direct writes.

**Element identifier**

direct\_write\_time

**Element type**

counter

Table 286. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Buffer Pool    |
| Table Space    | tablespace            | Buffer Pool    |

Table 286. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Buffer Pool    | bufferpool            | Buffer Pool    |
| Application    | appl                  | Buffer Pool    |

For snapshot monitoring, this counter can be reset.

Table 287. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Connection  | event_conn            | -              |
| Tablespaces | event_tablespace      | -              |

**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.

## Catalog cache monitor elements

The catalog cache stores:

- table descriptors for tables, views, and aliases. A descriptor stores information about a table, view, or alias in a condensed internal format. When an SQL statement references a table, it causes an insert of a table descriptor into the cache, so that subsequent SQL statements referencing that same table can use that descriptor and avoid reading from disk. (Operations reference a table descriptor when compiling an SQL statement.)
- database authorization information. Database authorization information is accessed during processing for statements like BIND, CONNECT, CREATE and LOAD. When a statement references database authorization information, subsequent operations referencing database authorization information for the same user or group can be accessed from the catalog cache, instead of from disk.
- execute privilege for routines, like user-defined functions and stored procedures. When a transaction references execute privilege for a particular routine, subsequent operations referencing the same routine can retrieve the information from the catalog cache instead of from disk.

The following database system monitor elements are used for catalog caches.

### **cat\_cache\_lookups - Catalog Cache Lookups**

The number of times that the catalog cache was referenced to obtain table descriptor information or authorization information.

**Element identifier**

cat\_cache\_lookups

**Element type**

counter

Table 288. 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 289. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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* 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.

See the *Administration Guide* for more information on the Catalog Cache Size configuration parameter.

### cat\_cache\_inserts - Catalog Cache Inserts

The number of times that the system tried to insert table descriptor or authorization information into the catalog cache.

#### Element identifier

cat\_cache\_inserts

#### Element type

counter

Table 290. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 290. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 291. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 `cat_cache_lookups` for more information on using this element.

### cat\_cache\_overflows - Catalog Cache Overflows

The number of times that the catalog cache overflowed the bounds of its allocated memory.

**Element identifier**

`cat_cache_overflows`

**Element type**

counter

Table 292. 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 293. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

### 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**

The largest size reached by the catalog cache.

**Note:** The **cat\_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.

#### **Element identifier**

cat\_cache\_size\_top

#### **Element type**

watermark

*Table 294. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 295. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

### **Usage**

This element indicates the maximum number of bytes the catalog cache required for the workload run against the database since it was activated.

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.

## **Package cache monitor elements**

The package and section information required for the execution of dynamic and static SQL statements are placed in the package cache as required. This information is required whenever a dynamic or static statement is being executed. The package cache exists at a database level. This means that agents with similar environments

can share the benefits of another agent's work. For static SQL statements, this can mean avoiding catalog access. For dynamic SQL statements, this can mean avoiding the cost of compilation.

The following database system monitor elements are used for package caches.

### **pkg\_cache\_lookups - Package Cache Lookups**

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.

#### **Element identifier**

pkg\_cache\_lookups

#### **Element type**

counter

*Table 296. 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 297. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

### **Usage**

To calculate the package cache miss ratio use the following formula:

$$1 - (\text{Package Cache Inserts} / \text{Package Cache Lookups})$$

The package cache miss ratio tells you whether or not the package cache is being used effectively. If the miss ratio is low (less than 0.2), the cache is performing well. A higher miss 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 *pckcachesz* 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 extra overhead incurred 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 on the quantity and type of sections being cached.

See the *Administration Guide* for more information on the Package Cache Size (*pckcachesz*) configuration parameter.

**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\_inserts - Package Cache Inserts**

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.

**Element identifier**

pkg\_cache\_inserts

**Element type**

counter

*Table 298. 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 299. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**Usage** In conjunction with "Package Cache Lookups", you can calculate the package cache hit ratio using the following formula:

$$1 - (\text{Package Cache Inserts} / \text{Package Cache Lookups})$$

See pkg\_cache\_lookups for information on using this element.

### **pkg\_cache\_num\_overflows - Package Cache Overflows**

The number of times that the package cache overflowed the bounds of its allocated memory.

**Element identifier**

pkg\_cache\_num\_overflows

**Element type**

counter

*Table 300. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 301. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

### **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.

**Element identifier**

pkg\_cache\_size\_top

**Element type**

watermark

*Table 302. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 303. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |



## 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:

$$\text{maximum package cache size} / 4096$$

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the package cache to avoid overflow.

## SQL workspaces monitor elements

**Note:** These monitor elements have been deprecated. Using these monitor elements will not generate an error. However, these monitor elements do not return a valid value. These monitor elements are no longer recommended and might be removed in a future release.

The following database system monitor elements are used for SQL workspaces.

### **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.

#### **Element identifier**

`shr_workspace_size_top`

#### **Element type**

water mark

*Table 304. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

*Table 305. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### 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.

**Element identifier**

shr\_workspace\_num\_overflows

**Element type**

counter

*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              | -              |
| Connection | event_conn            | -              |

**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\_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.

**Element identifier**

shr\_workspace\_section\_lookups

**Element type**

counter

*Table 308. 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 309. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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\_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.

**Element identifier**

shr\_workspace\_section\_inserts

**Element type**

counter

*Table 310. 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 311. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### **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.

**Element identifier**

priv\_workspace\_size\_top

**Element type**

water mark

Table 312. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

Table 313. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### **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.

**Element identifier**

priv\_workspace\_num\_overflows

**Element type**

counter

*Table 314. 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 315. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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\_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.

**Element identifier**

priv\_workspace\_section\_lookups

**Element type**

counter

*Table 316. 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 317. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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\_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.

#### Element identifier

priv\_workspace\_section\_inserts

#### Element type

counter

Table 318. 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 319. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### **appl\_section\_lookups - Section Lookups**

Lookups of SQL sections by an application from its shared SQL workspace.

**Element identifier**

appl\_section\_lookups

**Element type**

counter

*Table 320. 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 321. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

### **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\_section\_inserts - Section Inserts monitor element**

Inserts of SQL sections by an application from its shared SQL workspace.

**Element identifier**

appl\_section\_inserts

**Element type**

counter

*Table 322. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |
| Application    | appl                  | Basic          |

*Table 323. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

## 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.

## Database heap monitor elements

The following database system monitor elements are used for database heaps.

### **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.

#### **Element identifier**

db\_heap\_top

#### **Element type**

watermark

*Table 324. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 325. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## Logging monitor elements

The following database system monitor elements are used for logging.

### **sec\_log\_used\_top - Maximum Secondary Log Space Used**

The maximum amount of secondary log space used (in bytes).

#### **Element identifier**

sec\_log\_used\_top

#### **Element type**

watermark

*Table 326. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 327. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |



**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
- logretain

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.

For more information, see the *Administration Guide*.

**Note:** While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

### **tot\_log\_used\_top - Maximum Total Log Space Used**

The maximum amount of total log space used (in bytes).

**Element identifier**

tot\_log\_used\_top

**Element type**

watermark

*Table 328. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

*Table 329. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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

For more information, see the *Administration Guide*.

**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.

**Element identifier**

sec\_logs\_allocated

**Element type**

gauge

*Table 330. 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
- logretain

For more information, see the *Administration Guide*.

### 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 331. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 332. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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\_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 333. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 334. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

**uow\_log\_space\_used - Unit of Work Log Space Used**

The amount of log space (in bytes) used in the current unit of work of the monitored application.

**Element identifier**

uow\_log\_space\_used

**Element type**

gauge

*Table 335. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Unit of Work   |

*Table 336. Event Monitoring Information*

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Transactions | event_xact            | -              |

**Usage** You may use this element to understand the logging requirements at the unit of work level.

**total\_log\_used - Total Log Space Used**

The total amount of active log space currently used (in bytes) in the database.

**Element identifier**

total\_log\_used

**Element type**

gauge

Table 337. 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:

- `logfilsiz`
- `logprimary`
- `logsecond`

For more information, see the *Administration Guide*.

**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\_available - Total Log Available**

The amount of active log space in the database that is not being used by uncommitted transactions (in bytes).

**Element identifier**

`total_log_available`

**Element type**

gauge

Table 338. 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:

- `logfilsiz`
- `logprimary`
- `logsecond`

If `total_log_available` goes down to 0, `SQL0964N` will be returned. You may need to increase the above 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.

### **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 339. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 340. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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:

$$(\text{softmax} / 100) * \text{logfilsiz} * 4096$$

If this statement is not true, increase the number of page cleaners (*num\_iocleaners*) configuration parameter.

If the condition is true and it is desired that less log be held by dirty pages, then decrease the *softmax* configuration parameter.

### 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 341. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 342. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

### Element identifier

log\_write\_time

### Element type

time

Table 343. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 344. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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\_read\_time - Log Read Time

The total elapsed time spent by the logger reading log data from the disk.

### Element identifier

log\_read\_time

### Element type

time

Table 345. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 346. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

## 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.

### Element identifier

num\_log\_write\_io

### Element type

counter

Table 347. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 348. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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\_log\_read\_io - Number of Log Reads

The number of I/O requests issued by the logger for reading log data from the disk.

**Element identifier**

num\_log\_read\_io

**Element type**

counter

Table 349. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 350. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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\_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.

**Element identifier**

num\_log\_part\_page\_io

**Element type**

counter

Table 351. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 352. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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\_buffer\_full - Number of Full Log Buffers**

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.

**Element identifier**

num\_log\_buffer\_full

**Element type**

counter

*Table 353. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

**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.

**Element identifier**

num\_log\_data\_found\_in\_buffer

**Element type**

counter

*Table 354. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 355. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

### **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 356. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 357. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

### **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 358. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 359. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

### **current\_active\_log - Current Active Log File Number**

The file number of the active log file the DB2 database system is currently writing.

**Element identifier**  
current\_active\_log

**Element type**  
information

Table 360. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 361. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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.

**Element identifier**

current\_archive\_log

**Element type**

information

Table 362. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | detail_log            | Basic          |

Table 363. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

**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

---

## Database and application activity monitor elements

The following sections provide information on database and application activity.

### 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.

**Element identifier**

blocks\_pending\_cleanup

**Element type**

gauge

Table 364. 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.

## Locks and deadlocks monitor elements

The following elements provide information about locks and deadlocks.

### locks\_held - Locks Held

The number of locks currently held.

**Element identifier**

locks\_held

**Element type**

gauge

Table 365. 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 366. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |

**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.

### lock\_list\_in\_use - Total Lock List Memory In Use

The total amount of lock list memory (in bytes) that is in use.

**Element identifier**

lock\_list\_in\_use

**Element type**

watermark

Table 367. 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. See the *Administration Guide* for more information.

**Note:** When calculating utilization, it is important to note that the *locklist* configuration parameter is allocated in pages of 4K bytes each, while this monitor element provides results in bytes.

### **data\_partition\_id - Data partition identifier monitor element**

The identifier of the data partition for which information is returned.

**Element identifier**

data\_partition\_id

**Element type**

information

*Table 368. Snapshot monitoring information*

| Snapshot level | Logical data grouping | Monitor switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

*Table 369. 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.

When returning lock level information, a value of -1 represents a lock which controls access to the whole table. For non-partitioned tables, this element is absent from snapshots.

### **deadlocks - Deadlocks Detected**

The total number of deadlocks that have occurred.

**Element identifier**

deadlocks

**Element type**

counter

*Table 370. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 370. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |

For snapshot monitoring, this counter can be reset.

Table 371. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 execute 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 related elements listed above may also provide additional tuning suggestions.

## lock\_escals - Number of Lock Escalations

The number of times that locks have been escalated from several row locks to a table lock.

### Element identifier

lock\_escals

### Element type

counter

Table 372. 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 373. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Database     | event_db              | -              |
| Connection   | event_conn            | -              |
| Transactions | event_xact            | -              |

**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.

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.

To resolve these problems, you may be able to:

- Increase the *locklist* configuration parameter value. See the *Administration Guide* for a description of this configuration parameter.
- Increase the *maxlocks* configuration parameter value. See the *Administration Guide* for a description of this configuration parameter.
- Identify the applications with large numbers of locks (see *locks\_held\_top*), or those that are holding too much of the lock list, using the following formula:

$$(((locks\ held * 36) / (locklist * 4096)) * 100)$$

and comparing the value to *maxlocks*. 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*.

### **x\_lock\_escals - Exclusive Lock Escalations**

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.

#### **Element identifier**

x\_lock\_escals

#### **Element type**

counter

Table 374. 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 375. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Database     | event_db              | -              |
| Connection   | event_conn            | -              |
| Transactions | event_xact            | -              |

**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 *lock\_escals* 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.

## lock\_mode - Lock Mode

The type of lock being held.

### Element identifier

lock\_mode

### Element type

information

Table 376. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

Table 377. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |

Table 377. Event Monitoring Information (continued)

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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 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   | Next Key Share Lock                 | SQLM_LONS    |
| NX   | Next Key Exclusive Lock             | SQLM_LONX    |
| W    | Weak Exclusive Lock                 | SQLM_LOOW    |
| NW   | Next Key Weak Exclusive Lock        | SQLM_LONW    |

## lock\_status - Lock Status

Indicates the internal status of the lock.

### Element identifier

lock\_status

### Element type

information

Table 378. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 379. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |

**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:

**Granted state**

indicates that the application has the lock in the state specified by lock\_mode.

**Converting state**

indicates that the application is trying to change the lock held to a different type; for example, changing from a share lock to an exclusive lock.

**Note:** API users should refer to the *sqlmon.h* header file containing definitions of database system monitor constants.

**lock\_object\_type - Lock Object Type Waited On**

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).

**Element identifier**

lock\_object\_type

**Element type**

information

*Table 380. 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 381. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**Usage** This element can help you determine the source of contention for resources.

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.

## 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.

### Element identifier

lock\_object\_name

### Element type

information

Table 382. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |
| Lock           | lock                  | Basic          |

Table 383. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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\_node - Lock Node

The node involved in a lock.

### Element identifier

lock\_node

### Element type

information

Table 384. 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\_timeouts - Number of Lock Timeouts

The number of times that a request to lock an object timed-out instead of being granted.

### Element identifier

lock\_timeouts

### Element type

counter

Table 385. 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 386. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**Usage** This element can help you adjust the setting for the *locktimeout* database configuration parameter. If the number of lock time-outs 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 time-outs if your *locktimeout* database configuration parameter is set too high. In this case, your applications may wait excessively to obtain a lock. See the *Administration Guide* for more information.

## locks\_held\_top - Maximum Number of Locks Held

The maximum number of locks held during this transaction.

### Element identifier

locks\_held\_top

### Element type

counter

Table 387. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Transactions | event_xact            | -              |

**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. (See the *Administration Guide* for more information about this parameter.)

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 the following formula:

$$(\text{locklist} * 4096 / 36) * (\text{maxlocks} / 100)$$

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.

### **dl\_conns - Connections Involved in Deadlock**

The number of connections that are involved in the deadlock.

**Element identifier**

dl\_conns

**Element type**

gauge

*Table 388. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_deadlock        | -              |

**Usage** Use this element in your monitoring application to identify how many deadlock connection event records will follow in the event monitor data stream.

### **lock\_escalation - Lock Escalation**

Indicates whether a lock request was made as part of a lock escalation.

**Element identifier**

lock\_escalation

**Element type**

information

*Table 389. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Lock           |
| Lock           | lock_wait             | Lock           |

*Table 390. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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\_mode\_requested - Lock Mode Requested**

The lock mode being requested by the application.

**Element identifier**

lock\_mode\_requested

**Element type**

information

*Table 391. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock_wait             | Lock           |

*Table 392. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**Usage** The mode in which the lock was requested by the application. This value can help you determine the source of contention for resources.

**deadlock\_id - Deadlock Event Identifier**

The deadlock identifier for a deadlock.

**Element identifier**

deadlock\_id

**Element type**

information

*Table 393. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks                             | event_deadlock        | -              |
| 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_data_value      | -              |
| Deadlocks with Details History Values | event_detailed_dlconn | -              |
| Deadlocks with Details History Values | event_stmt_history    | -              |

**Usage** Use this element in your monitoring application to correlate deadlock connection and statement history event records with deadlock event records.

**deadlock\_node - Partition Number Where Deadlock Occurred**

Partition number where the deadlock occurred.

**Element identifier**

deadlock\_node

**Element type**  
information

*Table 394. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_deadlock        | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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.

### **participant\_no - Participant within Deadlock**

A sequence number uniquely identifying this participant within this deadlock.

**Element identifier**  
participant\_no

**Element type**  
information

*Table 395. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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 396. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**Usage** This element can help you determine which applications are in contention for resources.

### **rolled\_back\_participant\_no - Rolled Back Application Participant**

The participant number identifying the rolled back application.

**Element identifier**  
rolled\_back\_participant\_no

**Element type**  
information

*Table 397. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_deadlock        | -              |

**Usage** A system administrator can use this information to determine which application did not complete its updates, and determine which application should be started.

### **locks\_in\_list - Number of Locks Reported**

The number of locks held by a particular application to be reported on by the event monitor.

**Element identifier**  
locks\_in\_list

**Element type**  
information

*Table 398. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |

### **lock\_name - Lock Name**

Internal binary lock name. This element serves as a unique identifier for locks.

**Element identifier**  
lock\_name

**Element type**  
information

*Table 399. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | lock_wait      |

*Table 400. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

### **lock\_attributes - Lock Attributes**

Lock attributes.

**Element identifier**  
lock\_attributes

**Element type**  
information

Table 401. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Basic          |

Table 402. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

**Usage** The following are possible lock attribute settings. Each lock attribute setting is based upon a bit flag value defined in sqlmon.h.

| API Constant                  | Description              |
|-------------------------------|--------------------------|
| SQLM_LOCKATTR_WAIT_FOR_AVAIL  | Wait for availability.   |
| SQLM_LOCKATTR_ESCALATED       | Acquired by escalation.  |
| SQLM_LOCKATTR_RR_IN_BLOCK     | RR lock "in" block.      |
| SQLM_LOCKATTR_INSERT          | Insert lock.             |
| SQLM_LOCKATTR_DELETE_IN_BLOCK | Deleted row "in" block.  |
| SQLM_LOCKATTR_RR              | Lock by RR scan.         |
| SQLM_LOCKATTR_UPDATE_DELETE   | Update/delete row lock.  |
| SQLM_LOCKATTR_ALLOW_NEW       | Allow new lock requests. |
| SQLM_LOCKATTR_NEW_REQUEST     | A new lock requestor.    |

## lock\_release\_flags - Lock Release Flags

Lock release flags.

### Element identifier

lock\_release\_flags

### Element type

information

Table 403. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Basic          |

Table 404. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

**Usage** The following are possible release flag settings. Each release flag is based upon a bit flag value defined in sqlmon.h.

| API Constant                  | Description            |
|-------------------------------|------------------------|
| SQLM_LOCKRELFLAGS_SQLCOMPILER | Locks by SQL compiler. |



| API Constant                 | Description                  |
|------------------------------|------------------------------|
| SQLM_LOCKRELFIELDS_UNTRACKED | Non-unique, untracked locks. |

**Note:** All non-assigned bits are used for application cursors.

### lock\_count - Lock Count

The number of locks on the lock being held.

#### Element identifier

lock\_count

#### Element type

gauge

Table 405. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 406. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

**Usage** This value ranges from 1 to 255. It is incremented as new locks are acquired, and decremented as locks are released.

When lock\_count has a value of 255, this indicates that a *transaction duration lock* is being held. At this point, lock\_count is no longer incremented or decremented when locks are acquired or released. The lock\_count element is set to a value of 255 in one of two possible ways:

1. lock\_count 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\_hold\_count - Lock Hold Count

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.

#### Element identifier

lock\_hold\_count

#### Element type

gauge

Table 407. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |

Table 408. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

### **lock\_current\_mode - Original Lock Mode Before Conversion**

During a lock conversion operation, the type of lock held before the conversion is completed. The following is an example of a scenario that describes 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.

#### **Element identifier**

lock\_current\_mode

#### **Element type**

information

*Table 409. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock                  | Basic          |
| Lock           | lock_wait             | Basic          |

*Table 410. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |
| Deadlocks  | event_dlconn          | -              |

### **num\_indoubt\_trans - Number of Indoubt Transactions**

The number of outstanding indoubt transactions in the database.

#### **Element identifier**

num\_indoubt\_trans

#### **Element type**

gauge

*Table 411. 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.

## **Lock wait information monitor elements**

The following elements provide information that is returned when a DB2 agent working on behalf of an application is waiting to obtain a lock.

### **lock\_waits - Lock Waits**

The total number of times that applications or connections waited for locks.

**Element identifier**

lock\_waits

**Element type**

counter

*Table 412. 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 413. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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\_wait\_time - Time Waited On Locks**

The total elapsed time waited for a lock. Elapsed time is given in milliseconds.

**Element identifier**

lock\_wait\_time

**Element type**

counter

*Table 414. 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 415. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Database     | event_db              | -              |
| Connection   | event_conn            | -              |
| Transactions | event_xact            | -              |

**Usage** At the database level, this is the total amount of elapsed time that all applications were waiting for a lock within this database.

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 described above.

### **locks\_waiting - Current Agents Waiting On Locks**

Indicates the number of agents waiting on a lock.

**Element identifier**

locks\_waiting

**Element type**

gauge

Table 416. 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.

### **uow\_lock\_wait\_time - Total Time Unit of Work Waited on Locks**

The total amount of elapsed time this unit of work has spent waiting for locks.

**Element identifier**

uow\_lock\_wait\_time

**Element type**

counter

*Table 417. 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.

**lock\_wait\_start\_time - Lock Wait Start Timestamp**

The date and time that this application started waiting to obtain a lock on the object that is currently locked by another application.

**Element identifier**

lock\_wait\_start\_time

**Element type**

timestamp

*Table 418. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch  |
|----------------|-----------------------|-----------------|
| Application    | appl                  | Lock, Timestamp |
| Lock           | lock_wait             | Lock, Timestamp |

*Table 419. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | Timestamp      |
| Deadlocks with Details | event_detailed_dlconn | Timestamp      |

**Usage** This element can help you determine the severity of resource contention.

**lock\_timeout\_val - Lock timeout**

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.

**Element identifier**

lock\_timeout\_val

**Element type**

information

*Table 420. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Application    | agent                 | Basic          |

**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* value to see whether it is set too high inside the application. You can modify the application to lower the value of *lock\_timeout\_val* to let the application timeout, 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\_val* value is set too low and increase it as appropriate.

### **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.

**Element identifier**

agent\_id\_holding\_lock

**Element type**

information

*Table 421. 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.

### **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 422. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Lock           |
| Lock           | appl_lock_list        | Lock           |

Table 422. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Lock           | lock_wait             | Lock           |

Table 423. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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.

### 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 424. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |
| Lock           | appl_lock_list        | Basic          |

Table 425. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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.

### 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 426. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_deadlock        | -              |

**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\_agent\_id - Rolled Back Agent**

Agent that was rolled back when a deadlock occurred.

**Element identifier**  
rolled\_back\_agent\_id

**Element type**  
information

Table 427. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_deadlock        | -              |

**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\_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 428. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | event_deadlock        | -              |

**Usage** A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

## **Rollforward monitoring monitor elements**

Recovering database changes can be a time consuming process. You can use the database system monitor to monitor the progression of a recovery. The following elements provide information about rollforward status.

### **rf\_timestamp - Rollforward Timestamp**

The timestamp of the last committed transaction..



**Element identifier**  
rf\_timestamp

**Element type**  
timestamp

*Table 429. 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.

### **ts\_name - Tablespace Being Rolled Forward**

The name of the table space currently rolled forward.

**Element identifier**  
ts\_name

**Element type**  
information

*Table 430. 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.

### **rf\_type - Rollforward Type**

The type of rollforward in progress.

**Element identifier**  
rf\_type

**Element type**  
information

*Table 431. 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.

### **rf\_log\_num - Log Being Rolled Forward**

The log being processed.

**Element identifier**  
rf\_log\_num

**Element type**  
information

Table 432. 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.

### **rf\_status - Log Phase**

The status of the recovery.

**Element identifier**

rf\_status

**Element type**

information

Table 433. 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.

## **Table space activity monitor elements**

The following elements provide information about the table spaces.

### **tablespace\_id - Table Space Identification**

An integer that uniquely represents a table space used by the current database.

**Element identifier**

tablespace\_id

**Element type**

information

Table 434. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |
| Table          | table                 | Basic          |

Table 435. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**Usage** The value of this element matches a value from column TBSPACEID of view SYSCAT.TABLESPACES.

### **tablespace\_name - Table Space Name**

The name of a table space.

**Element identifier**

tablespace\_name

**Element type**

information

Table 436. 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 437. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |
| Table Space            | tablespace_list       | -              |

**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\_type - Table Space Type

The type of a table space.

#### Element identifier

tablespace\_type

#### Element type

information

Table 438. 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\_content\_type - Table Space Contents Type

The type of content in a table space.

**Element identifier**  
tablespace\_content\_type

**Element type**  
information

*Table 439. 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:

- 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\_state - Table Space State**

This element describes the current state of a table space.

**Element identifier**  
tablespace\_state

**Element type**  
information

*Table 440. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** 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. db2tbst - Get Tablespace State can be used to obtain the table space state associated with a given hexadecimal value.

*Table 441. 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                                      |

Table 441. Bit definitions listed in sqlutil.h (continued)

| Hexadecimal Value | Decimal Value | State                                               |
|-------------------|---------------|-----------------------------------------------------|
| 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                                        |
| 0x2000000         | 33554432      | Storage may be defined                              |
| 0x4000000         | 67108864      | Storage Definition is in 'final' state              |
| 0x8000000         | 134217728     | Storage Definition was changed prior to rollforward |
| 0x10000000        | 268435456     | DMS rebalancer is active                            |
| 0x20000000        | 536870912     | TBS deletion in progress                            |
| 0x40000000        | 1073741824    | TBS creation in progress                            |

### **tablespace\_page\_size - Table Space Page Size**

Page size used by a table space in bytes.

**Element identifier**

tablespace\_page\_size

**Element type**

information

Table 442. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

### **tablespace\_extent\_size - Table Space Extent Size**

The extent size used by a table space.

**Element identifier**

tablespace\_extent\_size

**Element type**

information

Table 443. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |

### **tablespace\_prefetch\_size - Table Space Prefetch Size**

The maximum number of pages the prefetcher gets from the disk at a time.

**Element identifier**

tablespace\_prefetch\_size

**Element type**  
information

Table 444. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace            | Basic          |
| Table Space    | tablespace_nodeinfo   | Basic          |

### Usage

- 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.
- 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\_cur\_pool\_id - Buffer Pool Currently Being Used**

The buffer pool identifier for a buffer pool that a table space is currently using.

**Element identifier**  
tablespace\_cur\_pool\_id

**Element type**  
information

Table 445. 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\_next\_pool\_id - Buffer Pool That Will Be Used at Next Startup**

The buffer pool identifier for a buffer pool that a table space will use at the next database startup.

**Element identifier**  
tablespace\_next\_pool\_id

**Element type**  
information

Table 446. 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\_total\_pages - Total Pages in Table Space**

Total number of pages in a table space.

**Element identifier**  
tablespace\_total\_pages

**Element type**  
information

Table 447. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch                                             |
|----------------|-----------------------|------------------------------------------------------------|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<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 (including overhead). 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\_usable\_pages - Usable Pages in Table Space**

The total number of pages in a table space minus overhead pages.

**Element identifier**  
tablespace\_usable\_pages

**Element type**  
information

Table 448. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch                                             |
|----------------|-----------------------|------------------------------------------------------------|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<br>Buffer Pool (SMS table spaces) |

**Usage** This element is applicable to DMS table spaces only. For SMS this element will have the same value as tablespace\_total\_pages.

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**

The total number of pages that are currently used (not free) in a table space.

**Element identifier**  
tablespace\_used\_pages

**Element type**  
information

Table 449. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch                                             |
|----------------|-----------------------|------------------------------------------------------------|
| Table Space    | tablespace_nodeinfo   | Basic (DMS table spaces)<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 tablespace\_total\_pages.

### tablespace\_free\_pages - Free Pages in Table Space

The total number of pages that are currently free in a table space.

**Element identifier**

tablespace\_free\_pages

**Element type**

information

Table 450. 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\_pending\_free\_pages - Pending Free Pages in Table Space

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.

**Element identifier**

tablespace\_pending\_free\_pages

**Element type**

information

Table 451. 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\_page\_top - Table space high watermark

The page in a table space that is holding the high watermark.

**Element identifier**

tablespace\_page\_top

**Element type**

watermark

Table 452. 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\_rebalancer\_mode - Rebalancer Mode**

An integer that represents whether a forward or reverse rebalance is taking place.

The tablespace\_rebalancer\_mode values (defined in sqlmon.h) are as follows:

- no rebalancing taking place: SQLM\_TABLESPACE\_NO\_REBAL
- forward: SQLM\_TABLESPACE\_FWD\_REBAL
- reverse: SQLM\_TABLESPACE\_REV\_REBAL

#### **Element identifier**

tablespace\_rebalancer\_mode

#### **Element type**

information

*Table 453. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

### **Usage**

This can be used as an indicator as to whether the current rebalance process is removing space from a table space or adding space to a table space. This is only applicable to a DMS table space.

### **tablespace\_rebalancer\_start\_time - Rebalancer Start Time**

A timestamp representing when a rebalancer was initially started.

#### **Element identifier**

tablespace\_rebalancer\_start\_time

#### **Element type**

information

*Table 454. 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\_restart\_time - Rebalancer Restart Time**

A timestamp representing when a rebalancer was restarted after being paused or stopped.

#### **Element identifier**

tablespace\_rebalancer\_restart\_time

#### **Element type**

information

Table 455. 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\_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).

**Element identifier**

tablespace\_rebalancer\_extents\_remaining

**Element type**

information

Table 456. 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 to check if rebalancing has completed. This is only applicable to a DMS table space.

### **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).

**Element identifier**

tablespace\_rebalancer\_extents\_processed

**Element type**

information

Table 457. 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\_last\_extent\_moved - Last Extent Moved by the Rebalancer**

The last extent moved by the rebalancer.

**Element identifier**

tablespace\_rebalancer\_last\_extent\_moved

**Element type**  
information

*Table 458. 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\_priority - Current Rebalancer Priority**

The priority at which the rebalancer is running in the database.

**Element identifier**  
`tablespace_rebalancer_priority`

**Element type**  
information

*Table 459. 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\_num\_quiescers - Number of Quiescers**

The number of users quiescing the table space (can be in the range of 0 to 5).

**Element identifier**  
`tablespace_num_quiescers`

**Element type**  
information

*Table 460. 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\_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 461. 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.

### **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 462. 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.

### **tablespace\_min\_recovery\_time - Minimum Recovery Time For Rollforward**

A timestamp showing the earliest point in time to which a table space can be rolled forward.

**Element identifier**

tablespace\_min\_recovery\_time

**Element type**

information

*Table 463. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

**Usage** Displayed only if non zero.

## tablespace\_num\_containers - Number of Containers in Table Space

Total number of containers in the table space.

### Element identifier

tablespace\_num\_containers

### Element type

information

Table 464. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

## 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.

### Element identifier

tablespace\_num\_ranges

### Element type

information

Table 465. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table Space    | tablespace_nodeinfo   | Basic          |

## fs\_caching - File System Caching

Indicates whether a particular table space uses file system caching.

### Element identifier

fs\_caching

### Element type

information

Table 466. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table space    | tablespace            | Basic          |

Table 467. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Tablespaces | event_tablespace      | -              |

## tablespace\_using\_auto\_storage - Using automatic storage

This element describes whether the table space was created as an automatic storage table space. A value of 1 means yes; 0 means no.

### Element identifier

tablespace\_using\_auto\_storage

### Element type

information

Table 468. 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.

### **tablespace\_auto\_resize\_enabled - Auto-resize enabled**

This element describes whether automatic resizing is enabled for the table space. A value of 1 means yes; 0 means no.

**Element identifier**

tablespace\_auto\_resize\_enabled

**Element type**

information

Table 469. 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 `tablespace_increase_size`, `tablespace_increase_size_percent`, and `tablespace_max_size` for the rate of increase and the maximum size for the table space.

### **tablespace\_initial\_size - Initial table space size**

The initial size of the automatic storage table space in bytes.

**Element identifier**

tablespace\_initial\_size

**Element type**

information

Table 470. 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\_current\_size - Current table space size**

This element shows the current size of the table space in bytes.

**Element identifier**

tablespace\_current\_size

**Element type**

information

Table 471. 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\_max\_size - Maximum table space size**

This element shows the maximum size in bytes to which the table space can automatically resize or increase.

**Element identifier**

`tablespace_max_size`

**Element type**

information

Table 472. 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\_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.

**Element identifier**

`tablespace_increase_size`

**Element type**

information

Table 473. 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**

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.

**Element identifier**

tablespace\_increase\_size\_percent

**Element type**

information

*Table 474. 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\_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.

**Element identifier**

tablespace\_last\_resize\_time

**Element type**

information

*Table 475. 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\_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.



**Element identifier**  
tablespace\_last\_resize\_failed

**Element type**  
information

*Table 476. 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.

### **Table space quiescer activity monitor elements**

The following elements provide information about table space quiescer activity.

#### **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 477. 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\_agent\_id - Quiescer Agent Identification:**

Agent ID of the agent holding a quiesce state.

**Element identifier**  
quiescer\_agent\_id

**Element type**  
information

*Table 478. 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\_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 479. 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.

#### **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 480. 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 481. 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.

### **Container status monitor elements**

The following elements provide information about container status.

#### **container\_id - Container Identification:**

An integer that uniquely defines a container within a table space.

**Element identifier**

container\_id

**Element type**

information

*Table 482. 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:**

The name of a container.

**Element identifier**

container\_name

**Element type**

information

*Table 483. 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\_type - Container Type:**

The type of the container.

**Element identifier**

container\_type

**Element type**

information

*Table 484. 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 values defined in sqlutil.h are as follows:

- Directory path (SMS): SQLB\_CONT\_PATH
- Raw device (DMS): SQLB\_CONT\_DISK

- File (DMS): SQLB\_CONT\_FILE
- Striped disk (DMS): SQLB\_CONT\_STRIPED\_DISK
- Striped file (DMS): SQLB\_CONT\_STRIPED\_FILE

**container\_total\_pages - Total Pages in Container:**

The total number of pages occupied by the container.

**Element identifier**

container\_total\_pages

**Element type**

information

*Table 485. 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\_usable\_pages - Usable Pages in Container:**

The total number of usable pages in a container.

**Element identifier**

container\_usable\_pages

**Element type**

information

*Table 486. 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.

**container\_stripe\_set - Stripe Set:**

The stripe set that a container belongs to.

**Element identifier**

container\_stripe\_set

**Element type**

information

Table 487. 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_accessible` to describe the container. This is only applicable to a DMS table space.

**container\_accessible - Accessibility of Container:**

This element describes if a container is accessible or not (1 meaning yes, 0 meaning no).

**Element identifier**

`container_accessible`

**Element type**

information

Table 488. 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.

A container may be inaccessible if it is being used by a process that needs the state of the table space to remain constant, such as an invocation of the LIST TABLESPACES command.

**Table space range status monitor elements**

The table space map is used to map logical table space page numbers to physical disk locations. The map is made up of a series of ranges.

For example, a range could look like this:

| Stripe | Range | MaxPage | MaxExtent | StartStripe | EndStripe | Adj | # Conts | Containers |
|--------|-------|---------|-----------|-------------|-----------|-----|---------|------------|
| 0      | [0]   | 249     | 124       | 0           | 124       | 0   | 1       | (0)        |
| 1      | [1]   | 999     | 499       | 125         | 249       | 0   | 3       | (0,1,2)    |
| 2      | [2]   | 1499    | 749       | 250         | 374       | 0   | 1       | (1,2)      |

A *container array* lists containers that belong to the range. The size of this array is determined by the total number of containers in the table space.

For each range, the following information will be returned in the snapshot.

**range\_stripe\_set\_number - Stripe Set Number:**

This value represents the stripe set in which a range resides.

**Element identifier**

`range_stripe_set_number`

**Element type**  
information

*Table 489. 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.

**Element identifier**  
range\_number

**Element type**  
information

*Table 490. 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.

**Element identifier**  
range\_max\_page\_number

**Element type**  
information

*Table 491. 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.

**Element identifier**  
range\_max\_extent

**Element type**  
information

*Table 492. 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.

**Element identifier**  
range\_start\_stripe

**Element type**  
information

*Table 493. 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.

**Element identifier**  
range\_end\_stripe

**Element type**  
information

*Table 494. 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\_adjustment - Range Adjustment:**

This value represents the offset into the container array in which a range actually starts.

**Element identifier**  
range\_adjustment

**Element type**  
information

*Table 495. 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.

**Element identifier**  
range\_num\_containers

**Element type**  
information

Table 496. 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.

**Element identifier**

range\_container\_id

**Element type**

information

Table 497. 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.

**Element identifier**

range\_offset

**Element type**

information

Table 498. 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.

## Table activity monitor elements

The following elements provide information about the tables.

**table\_type - Table Type**

The type of table for which information is returned.

**Element identifier**

table\_type

**Element type**

information

Table 499. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |



Table 500. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**Usage** You can 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:

- User table.
- User table that has been dropped.
- 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.
- System catalog table.

### **table\_name - Table Name**

The name of the table.

#### **Element identifier**

table\_name

#### **Element type**

information

Table 501. 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 502. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Tables                 | event_table           | -              |
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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 turned 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**

The schema of the table.

#### **Element identifier**

table\_schema

#### **Element type**

information

*Table 503. 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 504. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Tables                 | event_table           | -              |
| Deadlocks              | lock                  | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**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 turned 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 temp table
- *auth\_id* is the authorization ID used by the application to connect to the database

### rows\_deleted - Rows Deleted

This is the number of row deletions attempted.

**Element identifier**

rows\_deleted

**Element type**

counter

*Table 505. 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 506. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 *int\_rows\_deleted*.

### rows\_inserted - Rows Inserted

This is the number of row insertions attempted.

**Element identifier**

rows\_inserted

**Element type**

counter

*Table 507. 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 508. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 *int\_rows\_inserted*.

### rows\_updated - Rows Updated

This is the number of row updates attempted.

**Element identifier**

rows\_updated

**Element type**

counter

Table 509. 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 510. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 *int\_rows\_updated*. However, rows that are updated by more than one update statement are counted for each update.

### rows\_selected - Rows Selected

This is the number of rows that have been selected and returned to the application.

**Element identifier**

rows\_selected

**Element type**

counter

Table 511. 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 512. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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\_written - Rows Written

This is the number of rows changed (inserted, deleted or updated) in the table.

#### Element identifier

rows\_written

#### Element type

counter

Table 513. 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 514. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Connection   | event_conn            | -              |
| Tables       | event_table           | -              |
| Statements   | event_stmt            | -              |
| Transactions | event_xact            | -              |

**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.

### rows\_read - Rows Read

This is the number of rows read from the table.

#### Element identifier

rows\_read

#### Element type

counter

Table 515. 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 516. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Connection   | event_conn            | -              |
| Tables       | event_table           | -              |
| Statements   | event_stmt            | -              |
| Transactions | event_xact            | -              |

**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, you may use the SQL EXPLAIN statement, described in the *Administration Guide* 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 *overflow\_accesses*. 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 *rows\_read* is not incremented.

### **overflow\_accesses - Accesses to Overflowed Records**

The number of accesses (reads and writes) to overflowed rows of this table.

**Element identifier**

overflow\_accesses

**Element type**

counter

*Table 517. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 518. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

### **int\_rows\_deleted - Internal Rows Deleted**

This is the number of rows deleted from the database as a result of internal activity.

**Element identifier**

int\_rows\_deleted

**Element type**

counter

*Table 519. 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 520. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |
| Statements | event_stmt            | -              |

**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\_updated - Internal Rows Updated

This is the number of rows updated from the database as a result of internal activity.

**Element identifier**

int\_rows\_updated

**Element type**

counter

Table 521. 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 522. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |
| Statements | event_stmt            | -              |

**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.

### **int\_rows\_inserted - Internal Rows Inserted**

The number of rows inserted into the database as a result of internal activity caused by triggers.

**Element identifier**

int\_rows\_inserted

**Element type**

counter

*Table 523. 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 524. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |
| Statements | event_stmt            | -              |

**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.

### **table\_file\_id - Table File ID**

This is the file ID (FID) for the table.

**Element identifier**

table\_file\_id

**Element type**

information

*Table 525. 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 526. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Deadlocks  | lock                  | -              |

**Usage** 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* to identify the table.

## page\_reorgs - Page Reorganizations

The number of page reorganizations executed for a table.

### Element identifier

page\_reorgs

### Element type

counter

Table 527. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

For snapshot monitoring, this counter can be reset.

Table 528. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

## 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 reorgs.

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.

## 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, LOB data, and long data are reported by *index\_object\_pages*, *lob\_object\_pages*, and *long\_object\_pages*, respectively.

### Element identifier

data\_object\_pages

**Element type**  
information

*Table 529. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 530. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

### **index\_object\_pages - Index Object Pages**

The number of disk pages consumed by all indexes defined on a table.

**Element identifier**  
index\_object\_pages

**Element type**  
information

*Table 531. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

*Table 532. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

### **lob\_object\_pages - LOB Object Pages**

The number of disk pages consumed by LOB data.

**Element identifier**  
lob\_object\_pages

**Element type**  
information

*Table 533. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 534. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

### **long\_object\_pages - Long Object Pages**

The number of disk pages consumed by long data in a table.

**Element identifier**

long\_object\_pages

**Element type**

information

Table 535. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 536. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

### **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 537. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table                 | Basic          |

Table 538. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Tables     | event_table           | -              |

**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.

## Table reorganization monitor elements

The following elements provide information about table reorganization.

### reorg\_type - Table Reorganize Attributes

Table reorganize attribute settings.

**Element identifier**

reorg\_type

**Element type**

information

*Table 539. 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

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\_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.

**Element identifier**

reorg\_status

**Element type**

information

Table 540. 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

### reorg\_phase - Reorganize Phase

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.

**Element identifier**

reorg\_phase

**Element type**

information

Table 541. 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 sqlmon.h):

- 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 occurs on a non-partitioned index. The reorg\_phase element will indicate the Index Recreate phase only after the successful completion of all prior phases on every data partition.

### reorg\_phase\_start - Reorganize Phase Start Time

The start time of a phase of table reorganization. For partitioned tables, this will also indicate the start time of a reorganization phase for each data partition. During the index recreate phase, data groups for all data partitions are updated at the same time.

**Element identifier**

reorg\_phase\_start

**Element type**

timestamp

Table 542. 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 will occur during reorganization processing. This applies to classic (offline) reorganizations only. The range of values is 2 to 4 ([SORT], BUILD, REPLACE,[INDEX\_RECREATE]).

**Element identifier**

reorg\_max\_phase

**Element type**

information

Table 543. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

### 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.

**Element identifier**

reorg\_current\_counter

**Element type**

information

Table 544. 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\_max\_counter - Total Amount of Reorganization

A value that indicates the total amount of work to be done in a reorganization. This value can be used with reorg\_current\_counter, which represents the amount of work completed, to determine the progress of a reorganization.

**Element identifier**

reorg\_max\_counter

**Element type**

information

Table 545. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

## reorg\_completion - Reorganization Completion Flag

Table reorganization success indicator. For partitioned tables, this will also indicate the completion status for the data partition.

### Element identifier

reorg\_completion

### Element type

information

Table 546. 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 recreate fails on a partitioned table, the failed status is updated on all data partitions. See the administration notification log for further diagnostic information.

## reorg\_start - Table Reorganize Start Time

The start time of a table reorganization. For partitioned tables, this will also indicate the start time for each data partition reorganization.

### Element identifier

reorg\_start

### Element type

timestamp

Table 547. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

## reorg\_end - Table Reorganize End Time

The end time of a table reorganization. For partitioned tables, this will also indicate the end time for each data partition reorganization.

### Element identifier

reorg\_end

### Element type

timestamp

Table 548. 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.

**Element identifier**

reorg\_index\_id

**Element type**

information

*Table 549. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

### **reorg\_tbsp\_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.

**Element identifier**

reorg\_tbsp\_id

**Element type**

information

*Table 550. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Table          | table_reorg           | Basic          |

### **reorg\_long\_tbsp\_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.

**Element identifier**

reorg\_long\_tbsp\_id

**Element type**

information

*Table 551. 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.

**Element identifier**

reorg\_rows\_compressed

**Element type**

information

Table 552. 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.

**Element identifier**

reorg\_rows\_rejected\_for\_compression

**Element type**

information

Table 553. 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.

## SQL cursors monitor elements

The following elements provide information about the SQL cursors.

### 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 554. 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 555. 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*.

### **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 556. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

*Table 557. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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.

### **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 558. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

*Table 559. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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.

### **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 560. 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 561. 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*.

## SQL and XQuery statement activity monitor elements

The following elements provide information about SQL and XQuery statement activity.

### static\_sql\_stmts - Static SQL Statements Attempted

The number of static SQL statements that were attempted.

**Element identifier**

static\_sql\_stmts

**Element type**

counter

Table 562. 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 563. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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

```

## dynamic\_sql\_stmts - Dynamic SQL Statements Attempted

The number of dynamic SQL statements that were attempted.

### Element identifier

dynamic\_sql\_stmts

### Element type

counter

Table 564. 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 565. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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
```

## failed\_sql\_stmts - Failed Statement Operations

The number of SQL statements that were attempted, but failed.

### Element identifier

failed\_sql\_stmts

### Element type

counter

Table 566. 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 567. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

Table 567. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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.

### **commit\_sql\_stmts - Commit Statements Attempted**

The total number of SQL COMMIT statements that have been attempted.

#### **Element identifier**

commit\_sql\_stmts

#### **Element type**

counter

Table 568. 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 569. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 by calculating the sum of the following:

```

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.

### **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 570. 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 571. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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:

```

commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks

```



## select\_sql\_stmts - Select SQL Statements Executed

The number of SQL SELECT statements that were executed.

### Element identifier

select\_sql\_stmts

### Element type

counter

Table 572. 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 573. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

## uid\_sql\_stmts - Update/Insert/Delete SQL Statements Executed

The number of SQL UPDATE, INSERT, and DELETE statements that were executed.

### Element identifier

uid\_sql\_stmts

### Element type

counter

Table 574. 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 575. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### ddl\_sql\_stmts - Data Definition Language (DDL) SQL Statements

This element indicates the number of SQL Data Definition Language (DDL) statements that were executed.

**Element identifier**

ddl\_sql\_stmts

**Element type**

counter

Table 576. 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 577. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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:

$$\text{ddl\_sql\_stmts} / \text{total number of statements}$$

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 overhead to retrieve the information from the system catalogs

- the package cache, by invalidating sections that are stored there and causing additional system overhead due to section recompilation.

Examples of DDL statements are CREATE TABLE, CREATE VIEW, ALTER TABLE, and DROP INDEX.

### **int\_auto\_rebinds - Internal Automatic Rebinds**

The number of automatic rebinds (or recompiles) that have been attempted.

**Element identifier**

int\_auto\_rebinds

**Element type**

counter

*Table 578. 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 579. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**Usage** Automatic rebinds are the internal binds the system performs when an 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:

$$\text{int\_auto\_rebinds} / \text{total number of statements}$$

This information can be useful for analyzing application activity and throughput.

### **int\_commits - Internal Commits**

The total number of commits initiated internally by the database manager.

**Element identifier**

int\_commits

**Element type**

counter

Table 580. 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 581. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**Usage** An internal commit may occur during any of the following:

- 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 of the following:

```

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\_rollbacks - Internal Rollbacks

The total number of rollbacks initiated internally by the database manager.

**Element identifier**

int\_rollbacks

**Element type**

counter

Table 582. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

Table 582. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl                  | Basic          |

For snapshot monitoring, this counter can be reset.

Table 583. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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 `int_deadlock_rollbacks` 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\_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 584. 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 585. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Connection | event_conn            | -              |

**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`.

### **sql\_reqs\_since\_commit - SQL Requests Since Last Commit**

Number of SQL requests that have been submitted since the last commit.

**Element identifier**

`sql_reqs_since_commit`

**Element type**

information

Table 586. 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.

### **stmt\_node\_number - Statement Node**

Node where the statement was executed.

**Element identifier**

`stmt_node_number`

**Element type**

information

Table 587. 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.

### **binds\_precompiles - Binds/Precompiles Attempted**

The number of binds and pre-compiles attempted.

**Element identifier**

`binds_precompiles`

**Element type**

counter

Table 588. 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 589. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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.

### xquery\_stmts - XQuery Statements Attempted

The number of XQuery statements executed for an application or database.

#### Element identifier

xquery\_stmts

#### Element type

counter

Table 590. 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 591. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Connection | event_conn            | -              |

**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*.

## SQL statement details monitor elements

**Note:** Statement event monitors do not log fetches.

The following elements provide details about the SQL statements.

### stmt\_type - Statement Type

The type of statement processed.

**Element identifier**

stmt\_type

**Element type**

information

Table 592. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 593. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |
| Activities             | event_activitystmt    | -              |

**Usage** You can use this element to determine the type of statement that is executing. It can be one of the following:

- 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\_operation/operation - Statement Operation**

The statement operation currently being processed or most recently processed (if none currently running).

**Element identifier**

stmt\_operation (snapshot monitoring)

operation (event monitoring)

**Element type**

information

Table 594. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

Table 595. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |

**Usage** You can use this element to determine the operation that is executing or recently finished.



It can be one of the following.

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.

### **package\_name - Package Name**

The name of the package that contains the SQL statement currently executing.

#### **Element identifier**

package\_name

#### **Element type**

information

*Table 596. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

*Table 597. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |

Table 597. Event Monitoring Information (continued)

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activitystmt    | -              |

**Usage** You may use this element to help identify the application program and the SQL statement that is executing.

### consistency\_token - Package Consistency Token

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.

**Element identifier**

consistency\_token

**Element type**

information

Table 598. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 599. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | -              |

**Usage** You can use this element to help identify the package and the SQL statement that is executing.

### package\_version\_id - Package Version

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 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).

**Element identifier**

package\_version\_id

**Element type**

information

Table 600. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 601. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | -              |
| Activities | event_activitystmt    | -              |

**Usage** You may use this element to help identify the package and the SQL statement that is executing.

### **section\_number - Section Number**

The internal section number in the package for the SQL statement currently processing or most recently processed.

**Element identifier**

section\_number

**Element type**

information

*Table 602. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

*Table 603. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |
| Activities             | event_activitystmt    | -              |

**Usage** For static SQL, you can use this element along with creator, package\_version\_id, and package\_name 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
 PKGSHEMA = '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 contentions. Whenever possible, only use this query when there is little other activity against the database.

### **cursor\_name - Cursor Name**

The name of the cursor corresponding to this SQL statement.

**Element identifier**

cursor\_name

**Element type**

information

*Table 604. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 605. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |

**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.

### creator - Application Creator

The authorization ID of the user that pre-compiled the application.

#### Element identifier

creator

#### Element type

information

Table 606. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |

Table 607. 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.

### stmt\_start - Statement Operation Start Timestamp

The date and time when the stmt\_operation started executing.

#### Element identifier

stmt\_start

#### Element type

timestamp

Table 608. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | stmt                  | Statement, Timestamp |
| DCS Statement  | dc_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 609. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch       |
|----------------|-----------------------|----------------------|
| Application    | stmt                  | Statement, Timestamp |
| DCS Statement  | dc_stmt               | Statement, Timestamp |

**Usage** You can use this element with `stmt_start` to calculate the elapsed statement operation execution time.

### **stop\_time - Event Stop Time**

The date and time when the statement stopped executing.

**Element identifier**

`stop_time`

**Element type**

timestamp

Table 610. 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".

### **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.

**Element identifier**

`start_time`

**Element type**

timestamp

Table 611. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Database               | event_start           | Timestamp      |
| Transactions           | event_xact            | 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".

### **stmt\_elapsed\_time - Most Recent Statement Elapsed Time**

The elapsed execution time of the most recently completed statement.

**Element identifier**

stmt\_elapsed\_time

**Element type**

time

Table 612. 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.

### **insert\_timestamp - Statement insert timestamp monitor element**

The time when the statement was inserted into the cache.

**Element identifier**

insert\_timestamp

**Element type**

timestamp

Table 613. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

### **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.

### **stmt\_text - SQL Statement Text monitor element**

The text of the SQL statement.

**Element identifier**  
stmt\_text

**Element type**  
information

Table 614. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| Dynamic SQL    | dynsql                | Basic          |
| DCS Statement  | dcs_stmt              | Statement      |

Table 615. Event Monitoring Information

| Event Type                     | Logical Data Grouping | Monitor Switch |
|--------------------------------|-----------------------|----------------|
| Deadlocks with Details         | event_detailed_dlconn | -              |
| Deadlocks with Details History | event_stmt_history    | -              |
| Statements                     | event_stmt            | -              |
| Activities                     | event_activitystmt    | -              |

## 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 statements (event\_stmt) and deadlocks with details history (event\_stmt\_history) event monitors, this element is returned only for dynamic statements. For deadlocks with details (event\_detailed\_dlconn) and activities (event\_activitystmt) event monitors, **stmt\_text** is returned for dynamic and static statements only if it is available in the SQL statement cache. If an event monitor record cannot fit into the BUFFERSIZE of an event monitor, **stmt\_text** may be truncated so that the record can fit.

For information on 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\_sorts - Statement Sorts

The total number of times that a set of data was sorted in order to process the stmt\_operation.

**Element identifier**  
stmt\_sorts

**Element type**  
counter

Table 616. 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 above 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.

### **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.

**Element identifier**  
fetch\_count

**Element type**  
counter

Table 617. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |



Table 617. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Statement      |

For Dynamic SQL snapshot monitoring, this counter can be reset.

Table 618. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_stmt            | -              |

## 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.

## sqlca - SQL Communications Area (SQLCA)

The SQLCA data structure that was returned to the application at statement completion.

### Element identifier

sqlca

### Element type

information

Table 619. 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*.

## query\_card\_estimate - Query Number of Rows Estimate

An estimate of the number of rows that will be returned by a query.

### Element identifier

query\_card\_estimate

### Element type

information

Table 620. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |

Table 620. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| 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

Estimated cost, in timerons, for a query, as determined by the SQL compiler.

**Element identifier**

query\_cost\_estimate

**Element type**

information

Table 621. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcs_stmt              | Statement      |
| Activities     | event_activity        | -              |

**Usage** This 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® 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  $2^{*32} - 1$ .

### **stmt\_history\_id - Statement history identifier**

This numerical 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.

#### **Element identifier**

stmt\_history\_id

#### **Element type**

information

*Table 622. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | event_stmt_history    | -              |
| Deadlocks with Details<br>History Values | event_data_value      | -              |
| Deadlocks with Details<br>History        | event_stmt_history    | -              |

**Usage** You can use this information to see the sequence of SQL statements that caused the deadlock.

### **stmt\_first\_use\_time - Statement first use time**

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.

#### **Element identifier**

stmt\_first\_use\_time

#### **Element type**

information

*Table 623. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | event_stmt_history    | timestamp      |
| Deadlocks with Details<br>History        | event_stmt_history    | timestamp      |
| Activities                               | event_activitystmt    | timestamp      |

**Usage** Use this element in conjunction with other statement history entries to see the sequence of SQL statements that caused the deadlock.

### **stmt\_last\_use\_time - Statement last use time 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.

**Element identifier**

stmt\_last\_use\_time

**Element type**

information

*Table 624. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_stmt_history    | timestamp      |
| Deadlocks with Details History        | event_stmt_history    | timestamp      |
| Activities                            | event_activitystmt    | -              |

**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**

This element shows the lock timeout value in effect for the statement while it was being run.

**Element identifier**

stmt\_lock\_timeout

**Element type**

information

*Table 625. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_stmt_history    | -              |
| Deadlocks with Details History        | event_stmt_history    | -              |
| Activities                            | event_activitystmt    | -              |

**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\_isolation - Statement isolation**

This element shows the isolation value in effect for the statement while it was being run.

**Element identifier**

stmt\_isolation

**Element type**

information

*Table 626. Event Monitoring Information*

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_stmt_history    | -              |

Table 626. Event Monitoring Information (continued)

| Event Type                     | Logical Data Grouping | Monitor Switch |
|--------------------------------|-----------------------|----------------|
| Deadlocks with Details 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\_nest\_level - Statement nesting level

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).

#### Element identifier

stmt\_nest\_level

#### Element type

information

Table 627. Event Monitoring Information

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_stmt_history    | -              |
| Deadlocks with Details History        | event_stmt_history    | -              |
| Activities                            | event_activitystmt    | -              |

**Usage** You can use this element, along with stmt\_invocation\_id, to uniquely identify an invocation of a particular SQL statement. 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\_invocation\_id - Statement invocation identifier

This element shows the identifier (ID) of the routine invocation in which the SQL statement was run. The value indicates the number of routine invocations at the current nesting level that occurred while that level was active in the application.

#### Element identifier

stmt\_invocation\_id

#### Element type

information

Table 628. 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    | -              |

**Usage** You can use this element, along with `stmt_nest_level`, to uniquely identify an invocation of a particular SQL statement. 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\_query\_id - Statement query identifier**

This element shows the internal query identifier (ID) given to any SQL statement used as a cursor.

**Element identifier**

stmt\_query\_id

**Element type**

information

Table 629. 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    | -              |

**Usage** You can use this element, along with `stmt_nest_level`, 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\_source\_id - Statement source identifier**

This element shows the internal identifier (ID) given to the source of the SQL statement that was run.

**Element identifier**

stmt\_source\_id

**Element type**

information

Table 630. 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    | -              |

## Usage

You can use this element, along with `appl_id`, 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\_pkgcache\_id - Statement package cache identifier**

This element shows the internal package cache identifier (ID) for a dynamic SQL statement.

#### Element identifier

`stmt_pkgcache_id`

#### Element type

information

*Table 631. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

*Table 632. 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    | -              |

## 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.

### **comp\_env\_desc - Compilation environment handle**

This element represents a handle to the compilation environment used when compiling the SQL statement.

#### Element identifier

`comp_env_desc`

#### Element type

information

*Table 633. 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    | -              |

**Usage** You can provide this element as input to the COMPILATION\_ENV table function, or to the SET COMPILATION ENVIRONMENT SQL statement.

### stmt\_value\_type - Value type

This element contains a string representation of the type of a data value associated with an SQL statement.

**Element identifier**

stmt\_value\_type

**Element type**

information

*Table 634. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | stmt_value_type       | -              |
| Activities                               | event_activityvals    | -              |

**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

This element shows whether a data value associated with an SQL statement is the NULL value.

**Element identifier**

stmt\_value\_isnull

**Element type**

information

*Table 635. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | stmt_value_isnull     | -              |
| Activities                               | event_activityvals    | -              |

**Usage** You can use this element in conjunction with other statement history entries to understand the cause of the deadlock.

### 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.

**Element identifier**

stmt\_value\_data

**Element type**

information

*Table 636. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | stmt_value_data       | -              |
| Activities                               | event_activityvals    | -              |



**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.

**Element identifier**

stmt\_value\_index

**Element type**

information

*Table 637. Event Monitoring Information*

| Event Type                               | Logical Data Grouping | Monitor Switch |
|------------------------------------------|-----------------------|----------------|
| Deadlocks with Details<br>History Values | stmt_value_data       | -              |
| Activities                               | event_activityvals    | -              |

**Usage** You can use this element in conjunction with other statement history entries to understand the cause of 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.

**Element identifier**

inact\_stmthist\_sz

**Element type**

information

*Table 638. 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\_value\_isreopt - Variable used for statement reoptimization**

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.

**Element identifier**

stmt\_value\_isreopt

**Element type**

information

Table 639. Event Monitoring Information

| Event Type                            | Logical Data Grouping | Monitor Switch |
|---------------------------------------|-----------------------|----------------|
| Deadlocks with Details History Values | event_data_value      | -              |
| Activities                            | event_activityvals    | -              |

## 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.

## Subsection details monitor elements

When a statement is executed against a partitioned database, it is divided into subsections that may be executed on different partitions. An application may have several subsections simultaneously executing on a partition.

For problem determination, you may have to locate the problem subsection. For example, a subsection may be waiting on a tablequeue, because one of the writers to this tablequeue is in lock wait on another node. To get the overall picture for an application, you may have to issue an application snapshot on each node where the application is running.

The following database system monitor elements provide information about subsections.

### ss\_number - Subsection Number

Identifies the subsection associated with the returned information.

#### Element identifier

ss\_number

#### Element type

information

Table 640. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

Table 641. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** This number relates to the subsection number in the access plan that can be obtained with db2expln.

### ss\_node\_number - Subsection Node Number

Node where the subsection was executed.

#### Element identifier

ss\_node\_number

**Element type**  
information

*Table 642. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 643. 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\_status - Subsection Status**

The current status of an executing subsection.

**Element identifier**  
ss\_status

**Element type**  
information

*Table 644. 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 tablequeue
- waiting to send data on a tablequeue

### **ss\_exec\_time - Subsection Execution Elapsed Time**

The time in seconds that it took a subsection to execute.

**Element identifier**  
ss\_exec\_time

**Element type**  
counter

*Table 645. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 646. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** Allows you to track the progress of a subsection.

## **tq\_wait\_for\_any - Waiting for Any Node to Send on a Tablequeue**

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 647. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** If *ss\_status* indicates *waiting to receive data on a tablequeue* 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.

## **tq\_node\_waited\_for - Waited for Node on a Tablequeue**

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 648. 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\_tot\_send\_spills - Total Number of Tablequeue Buffers Overflowed**

Total number of tablequeue buffers overflowed to a temporary table.

### **Element identifier**

tq\_tot\_send\_spills

### **Element type**

counter

Table 649. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

Table 650. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** Indicates the total number of tablequeue buffers that have been written to a temporary table. See `tq_cur_send_spills` for more information.

### **tq\_cur\_send\_spills - Current Number of Tablequeue Buffers Overflowed**

Current number of tablequeue buffers residing in a temporary table.

**Element identifier**

`tq_cur_send_spills`

**Element type**

gauge

Table 651. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** An agent writing to a tablequeue 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 `db2diad.log` indicating that you ran out of temporary space in the TEMP table space, then tablequeue 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\_rows\_read - Number of Rows Read from Tablequeues**

Total number of rows read from tablequeues.

**Element identifier**

`tq_rows_read`

**Element type**

counter

Table 652. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

Table 653. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**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 Tablequeues**

Total number of rows written to tablequeues.

**Element identifier**

tq\_rows\_written

**Element type**

counter

*Table 654. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 655. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**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\_max\_send\_spills - Maximum Number of Tablequeue Buffers Overflows**

Maximum number of tablequeue buffers overflowed to a temporary table.

**Element identifier**

tq\_max\_send\_spills

**Element type**

watermark

*Table 656. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

*Table 657. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statements | event_subsection      | -              |

**Usage** Indicates the maximum number of tablequeue buffers that have been written to a temporary table.

## **tq\_id\_waiting\_on - Waited on Node on a Tablequeue**

The agent that is waiting.

### **Element identifier**

tq\_id\_waiting\_on

### **Element type**

information

*Table 658. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Statement      |

**Usage** This can be used for troubleshooting.

## **Dynamic SQL monitor elements**

The DB2 statement cache stores packages and statistics for frequently used SQL statements. By examining the contents of this cache, you can identify the dynamic SQL statements that are most frequently executed, and the queries that consume the most resource. Using this information, you can examine the most commonly executed and most expensive SQL operations, to determine if SQL tuning could result in better database performance.

### **num\_executions - Statement Executions**

The number of times that an SQL statement has been executed.

#### **Element identifier**

num\_executions

#### **Element type**

counter

*Table 659. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Dynamic SQL    | dynsql                | Basic          |

For snapshot monitoring, this counter can be reset.

**Usage** You can use this element to identify the most frequently executed SQL statements in your system.

### **num\_compilations - Statement Compilations**

The number of different compilations for a specific SQL statement.

#### **Element identifier**

num\_compilations

#### **Element type**

counter

*Table 660. 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 foo" 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.

### **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.

**Element identifier**

`prep_time_worst`

**Element type**

watermark

*Table 661. 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.

### **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.

**Element identifier**

`prep_time_best`

**Element type**

water mark

*Table 662. 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.

### **total\_exec\_time - Elapsed Statement Execution Time**

The total time in seconds and microseconds that was spent executing a particular statement in the SQL cache.

**Element identifier**

`total_exec_time`

**Element type**

time



Table 663. 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 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 must be considered when evaluating the contents of this element.

## Intra-query parallelism monitor elements

The following database system monitor elements provide information about queries for which the degree of parallelism is greater than 1.

### 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 664. 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.

### 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 665. 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.

### degree\_parallelism - Degree of Parallelism

The degree of parallelism requested when the query was bound.

**Element identifier**

degree\_parallelism

**Element type**  
information

Table 666. 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.

## CPU usage monitor elements

The CPU usage for an application is broken down into *user CPU*, which is the CPU consumed while executing application code, and *system CPU*, which is the CPU consumed executing system calls.

CPU consumption is available at the application, transaction, statement, and subsection levels.

### **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 667. 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\_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 668. 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.

### 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 669. 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\_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 670. 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.

### **user\_cpu\_time - User CPU Time**

The total *user* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

When either the statement monitor switch or the timestamp switch is not turned on, this element is not collected and -1 is written instead.

**Element identifier**

user\_cpu\_time

**Element type**

time

Table 671. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Connection   | event_conn            | -              |
| Transactions | event_xact            | -              |
| Statements   | event_stmt            | -              |
| Activities   | event_activity        | -              |

**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.

### **system\_cpu\_time - System CPU Time**

The total *system* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

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.

**Element identifier**

system\_cpu\_time

**Element type**  
time

Table 672. Event Monitoring Information

| Event Type   | Logical Data Grouping | Monitor Switch |
|--------------|-----------------------|----------------|
| Connection   | event_conn            | -              |
| Transactions | event_xact            | -              |
| Statements   | event_stmt            | -              |
| Activities   | event_activity        | -              |

**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.

### **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.

**Element identifier**  
ss\_usr\_cpu\_time

**Element type**  
time

Table 673. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Timestamp      |

Table 674. 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\_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.

**Element identifier**  
ss\_sys\_cpu\_time

**Element type**  
time

Table 675. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | subsection            | Timestamp      |

Table 676. 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.

### **total\_sys\_cpu\_time - Total System CPU for a Statement**

The total system CPU time for an SQL statement.

**Element identifier**

total\_sys\_cpu\_time

**Element type**

time

Table 677. 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.

### **total\_usr\_cpu\_time - Total User CPU for a Statement**

The total user CPU time for an SQL statement.

**Element identifier**

total\_usr\_cpu\_time

**Element type**

time

Table 678. 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.

## **Snapshot monitoring monitor elements**

The following elements provide information about monitoring applications. They are returned as output for every snapshot.

## 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 679. 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.

## input\_db\_alias - Input Database Alias

The alias of the database provided when calling the snapshot function.

### Element identifier

input\_db\_alias

### Element type

information

Table 680. 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.

### **time\_stamp - Snapshot Time**

The date and time when the database system monitor information was collected.

**Element identifier**  
time\_stamp

**Element type**  
timestamp

*Table 681. 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.

### **num\_nodes\_in\_db2\_instance - Number of Nodes in Partition**

The number of nodes on the instance where the snapshot was taken.

**Element identifier**  
num\_nodes\_in\_db2\_instance

**Element type**  
information

*Table 682. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |

*Table 683. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

**Usage** Use this element to determine the number of nodes for an instance. For non-partitioned system databases, this value will be 1.

## **Event monitoring monitor elements**

The following elements provide information about monitoring applications. They are returned as output for events.

### **count - Number of Event Monitor Overflows**

The number of consecutive overflows that have occurred.

**Element identifier**  
count

**Element type**  
counter

*Table 684. Event Monitoring Information*

| Event Type      | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Overflow Record | event_overflow        | -              |

**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.

### **first\_overflow\_time - Time of First Event Overflow**

The date and time of the first overflow recorded by this overflow record.

**Element identifier**

first\_overflow\_time

**Element type**

timestamp

*Table 685. 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.

### **last\_overflow\_time - Time of Last Event Overflow**

The date and time of the last overflow recorded this overflow record.

**Element identifier**

last\_overflow\_time

**Element type**

timestamp

*Table 686. 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.

### **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).

**Element identifier**

byte\_order

**Element type**

information

*Table 687. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

**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

### **version - Version of Monitor Data**

The version of the database manager that produced the event monitor data stream.

#### **Element identifier**

version

#### **Element type**

information

*Table 688. 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.

### **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 689. Event Monitoring Information*

| Event Type       | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Event Log Header | event_log_header      | -              |

**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.

### **partial\_record - Partial Record monitor element**

Indicates that an event monitor record is only a partial record.

#### **Element identifier**

partial\_record

#### **Element type**

information

Table 690. 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_activitystmt`, or `event_activityvals` records.

## event\_time - Event Time

The date and time an event occurred.

**Element identifier**  
event\_time

**Element type**  
information

Table 691. 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.

## 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 692. Event Monitoring Information

| Event Type  | Logical Data Grouping | Monitor Switch |
|-------------|-----------------------|----------------|
| Database    | event_db              | -              |
| Tables      | event_table           | -              |
| Tablespaces | event_tablespace      | -              |
| Bufferpools | event_bufferpool      | -              |

**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.

## evmon\_activates - Number of Event Monitor Activations

The number of times an event monitor has been activated.

### Element identifier

evmon\_activates

### Element type

counter

Table 693. Event Monitoring Information

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Database               | event_db              | -              |
| Tables                 | event_table           | -              |
| Tablespaces            | event_tablespace      | -              |
| Bufferpools            | event_bufferpool      | -              |
| Deadlocks              | event_deadlock        | -              |
| Deadlocks              | event_dlconn          | -              |
| Deadlocks with Details | event_detailed_dlconn | -              |

**Usage** Use this element to correlate information returned by the above event types. This element is applicable only to write-to-table event monitors. This monitor element is not maintained for event monitors that write to a file or pipe.

Only some types of write-to-table event monitors use the evmon\_activates monitor element (the event monitor types that do use this element are listed in the previous table, "Event Monitoring Information"). These event monitors update the evmon\_activates column of the SYSCAT.EVENTMONITORS catalog table when activated. This change is logged, so the DATABASE CONFIGURATION will display:

Database is consistent = 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 will be produced.

### sql\_req\_id - Request Identifier for SQL Statement

The request identifier for an operation in an SQL statement.

**Element identifier**

sql\_req\_id

**Element type**

information

*Table 694. 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.

### 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.

**Element identifier**

message

**Element type**

information

*Table 695. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| -          | -                     | -              |

### Usage

The following are possible values:

**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.

**Element identifier**  
message\_time

**Element type**  
timestamp

*Table 696. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| -          | -                     | -              |

### **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. This value indicates the number of the partition where event monitor data is inserted.

**Element identifier**  
partition\_number

**Element type**  
information

*Table 697. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| -          | -                     | -              |

## **Utilities monitor elements**

The following elements provide information about utilities.

### **utility\_dbname - Database Operated on by Utility**

The database operated on by the utility.

**Element identifier**  
utility\_dbname

**Element type**  
information

*Table 698. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

### **utility\_id - Utility ID**

The unique identifier corresponding to the utility invocation.

**Element identifier**  
utility\_id

**Element type**  
information

Table 699. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

### utility\_type - Utility Type

The class of utility.

**Element identifier**  
utility\_type

**Element type**  
information

Table 700. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

### Usage

The values for this element can be any of the constants defined in sqlmon.h with names beginning "SQLM\_UTILITY\_".

### 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.

**Element identifier**  
utility\_priority

**Element type**  
information

Table 701. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

### utility\_start\_time - Utility Start Time

The date and time when the current utility was originally invoked.

**Element identifier**  
utility\_start\_time

**Element type**  
timestamp

Table 702. 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.

**Element identifier**

utility\_description

**Element type**

information

*Table 703. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

**utility\_state - Utility State**

This element describes the state of a utility.

**Element identifier**

utility\_state

**Element type**

information

*Table 704. 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                                              |
| 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\_invoker\_type - Utility Invoker Type**

This element describes how a utility was invoked.

**Element identifier**

utility\_invoker\_type

**Element type**

information

*Table 705. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | utility_info          | Basic          |

**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 |



| API Constant              | Utility                                  |
|---------------------------|------------------------------------------|
| SQLM_UTILITY_INVOKER_AUTO | Utility was invoked automatically by DB2 |

### **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.

**Element identifier**

progress\_list\_cur\_seq\_num

**Element type**

information

*Table 706. 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.

**Element identifier**

progress\_seq\_num

**Element type**

information

*Table 707. 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\_description - Progress Description**

Describes the phase of work.

**Element identifier**

progress\_description

**Element type**

information

*Table 708. 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\_start\_time - Progress Start Time**

A timestamp representing the start of the phase.

**Element identifier**

progress\_start\_time

**Element type**

information

*Table 709. 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\_work\_metric - Progress Work Metric**

The metric for interpreting the *progress\_total\_units* and *progress\_completed\_units* elements.

**Element identifier**

progress\_work\_metric

**Element type**

information

*Table 710. 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.

### **progress\_total\_units - Total Progress Work Units**

Total amount of work to perform in order for the phase to be complete.

**Element identifier**

progress\_total\_units

**Element type**  
information

Table 711. 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\_completed\_units - Completed Progress Work Units**

The number of work units for the current phase which have been completed.

**Element identifier**  
*progress\_completed\_units*

**Element type**  
information

Table 712. 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:

$$\text{percentage complete} = \text{progress\_completed\_units} / \text{progress\_total\_units} * 100$$

### **progress\_list\_attr - Current Progress List Attributes**

This element describes how to interpret a list of progress elements.

**Element identifier**  
*progress\_list\_attr*

**Element type**  
information

Table 713. 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.

---

## High availability disaster recovery (HADR) monitor elements

The DB2 database high availability disaster recovery (HADR) is a database replication feature that provides a high availability solution for both partial and complete site failures.

HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. When a failure occurs on the primary, you can fail over to the standby. The standby then becomes the new primary. Since the standby database server is already online, failover can be accomplished very quickly, resulting in minimal down time.

The following monitor elements allow you to examine the current configuration and state of the HADR subsystem.

### hadr\_role - HADR Role

The current high availability disaster recovery (HADR) role of the database.

**Element identifier**  
hadr\_role

**Element type**  
information

Table 714. 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.

## **hadr\_state - HADR State monitor element**

The current high availability disaster recovery (HADR) state of the database.

### **Element identifier**

hadr\_state

### **Element type**

information

*Table 715. 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.

## hadr\_syncmode - HADR Synchronization Mode monitor element

The current high availability disaster recovery (HADR) synchronization mode of the database.

**Element identifier**

hadr\_syncmode

**Element type**

information

*Table 716. 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.

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 the HADR system is actually using rather than the value in the database configuration file.

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.

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\_status - HADR Connection Status monitor element

The current high availability disaster recovery (HADR) connection status of the database.

**Element identifier**

hadr\_connect\_status

**Element type**

information

*Table 717. 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.

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

Shows one of the following: high availability disaster recovery (HADR) connection time, HADR congestion time, or HADR disconnection time.

### Element identifier

hadr\_connect\_time

### Element type

timestamp

Table 718. 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.

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 missed heartbeats on the high availability disaster recovery (HADR) connection. If the database is in HADR primary or standby role, this element indicates the health of the HADR connection.

### Element identifier

hadr\_heartbeat

### Element type

counter

*Table 719. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

For snapshot monitoring, this counter cannot be reset.

## Usage

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.

An HADR database expects at least one heartbeat message from the other database in each quarter of the time interval defined in the `HADR_TIMEOUT` database configuration parameter, or in 30 seconds, whichever is shorter. For example, if the `HADR_TIMEOUT` value is 80 (seconds), then the HADR database expects at least one heartbeat message from the other database every 20 seconds.

The data type of this element is integer.

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\_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".

### Element identifier

hadr\_local\_host



**Element type**  
information

Table 720. 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.

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.

**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.

## 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.

**Element identifier**  
hadr\_local\_service

**Element type**  
information

Table 721. 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. 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 the HADR system is actually using rather than the value in the database configuration file.

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".

**Element identifier**  
hadr\_remote\_host

**Element type**  
information

Table 722. 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. 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 the HADR system is actually using rather than the value in the database configuration file.

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.

**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.

## 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.

### Element identifier

hadr\_remote\_service

### Element type

information

Table 723. 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. 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 the HADR system is actually using rather than the value in the database configuration file.

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.

### Element identifier

hadr\_remote\_instance

### Element type

information

Table 724. 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. 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 the HADR system is actually using rather than the value in the database configuration file.

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.

### Element identifier

hadr\_timeout

### Element type

information

Table 725. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | hadr                  | Basic          |

## Usage

Use this element to determine the effective HADR timeout value. 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 the HADR system is actually using rather than the value in the database configuration file.

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\_file - HADR Primary Log File monitor element

The name of the current log file on the primary HADR database.

### Element identifier

hadr\_primary\_log\_file

### Element type

information

Table 726. 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.

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.

### Element identifier

hadr\_primary\_log\_page

### Element type

information

Table 727. 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.

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.

### Element identifier

hadr\_primary\_log\_lsn

### Element type

information

Table 728. 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.

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\_file - HADR Standby Log File monitor element**

The name of the current log file on the standby HADR database.

### Element identifier

hadr\_standby\_log\_file

### Element type

information

*Table 729. 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.

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.

### Element identifier

hadr\_standby\_log\_page

### Element type

information

*Table 730. 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.

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.

### Element identifier

hadr\_standby\_log\_lsn

### Element type

information

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.

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.

### Element identifier

hadr\_log\_gap

### Element type

information

Table 732. 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.

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.

**Element identifier**  
hadr\_peer\_window

**Element type**  
information

*Table 733. 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.

**Element identifier**  
hadr\_peer\_window\_end

**Element type**  
timestamp

*Table 734. 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.

---

## DB2 Connect monitor elements

The following elements provide DB2 Connection information at the database, application, transaction, and statement levels.

### dc\_s\_db\_name - DCS Database Name

The name of the DCS database as cataloged in the DCS directory.

**Element identifier**

dc\_s\_db\_name

**Element type**

information

*Table 735. 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.

### 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.

**Element identifier**

host\_db\_name

**Element type**

information

*Table 736. 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.

### 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 737. 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.



## 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.

**Element identifier**

gw\_con\_time

**Element type**

timestamp

*Table 738. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Database    | dcс_dbase             | Timestamp      |
| DCS Application | dcс_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 739. 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\_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.

**Element identifier**

gw\_total\_cons

**Element type**

watermark

*Table 740. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcс_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.

## **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.

**Element identifier**

gw\_cur\_cons

**Element type**

gauge

*Table 741. 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\_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 742. Snapshot Monitoring Information*

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dcs_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\_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 743. Snapshot Monitoring Information

| Snapshot Level   | Logical Data Grouping | Monitor Switch |
|------------------|-----------------------|----------------|
| Database Manager | db2                   | Basic          |
| DCS Database     | dc_s_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\_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.

**Element identifier**

gw\_exec\_time

**Element type**

time

Table 744. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch       |
|-----------------|-----------------------|----------------------|
| DCS Application | dc_s_appl             | Statement, Timestamp |
| DCS Statement   | dc_s_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.

## 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.

**Element identifier**

sql\_stmts

**Element type**

counter

Table 745. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Basic          |
| DCS Application   | dc_s_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.

## 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.

**Element identifier**

sql\_chains

**Element type**

counter

Table 746. 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.

## open\_cursors - Number of Open Cursors

The number of cursors currently open for an application.

**Element identifier**

open\_cursors

**Element type**

gauge

Table 747. 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 RQRI0BLK. 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.

## dcs\_appl\_status - DCS Application Status

The status of a DCS application at the DB2 Connect gateway.

**Element identifier**

dcs\_appl\_status

**Element type**

information

Table 748. 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. 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.

## 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 749. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcs_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.

## 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 750. 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.

## outbound\_comm\_protocol - Outbound Communication Protocol

The communication protocol used between the DB2 Connect gateway and the host.

**Element identifier**  
outbound\_comm\_protocol

**Element type**  
information

Table 751. 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. The valid value is:

- SQLM\_PROT\_TCPIP

## outbound\_comm\_address - Outbound Communication Address

This is the communication address of the target database. For example, it could be an SNA net ID and LU partner name, or an IP address and port number for TCP/IP.

### Element identifier

outbound\_comm\_address

### Element type

information

Table 752. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dc_s_appl_info        | Basic          |

**Usage** Use this element for problem determination on DCS applications.

## inbound\_comm\_address - Inbound Communication Address

This is the communication address of the client. For example, it could be an SNA net ID and LU partner name, or an IP address and port number for TCP/IP.

### Element identifier

inbound\_comm\_address

### Element type

information

Table 753. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dc_s_appl_info        | Basic          |

**Usage** Use this element for problem determination on DCS applications.

## inbound\_bytes\_received - Inbound Number of Bytes Received

The number of bytes received by the DB2 Connect gateway from the client, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

### Element identifier

inbound\_bytes\_received

**Element type**  
counter

*Table 754. 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.

## **outbound\_bytes\_sent - Outbound Number of Bytes Sent**

The number of bytes sent by the DB2 Connect gateway to the host, excluding communication protocol overhead (for example, TCP/IP or SNA headers). 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.

**Element identifier**  
outbound\_bytes\_sent

**Element type**  
counter

*Table 755. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dc_s_dbase            | Basic          |
| DCS Application   | dc_s_appl             | Basic          |
| DCS Statement     | dc_s_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\_received - Outbound Number of Bytes Received**

The number of bytes received by the DB2 Connect gateway from the host, excluding communication protocol overhead (for example, TCP/IP or SNA headers). 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.

**Element identifier**  
outbound\_bytes\_received

**Element type**  
counter



Table 756. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcс_dbase             | Basic          |
| DCS Application   | dcс_appl              | Basic          |
| DCS Statement     | dcс_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 host databases 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 overhead (for example, TCP/IP or SNA headers).

### Element identifier

inbound\_bytes\_sent

### Element type

counter

Table 757. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dcс_appl              | Basic          |
| DCS Statement   | dcс_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.

## 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.

### Element identifier

outbound\_bytes\_sent\_top

### Element type

watermark

Table 758. 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\_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.

**Element identifier**  
outbound\_bytes\_received\_top

**Element type**  
watermark

*Table 759. 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\_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.

**Element identifier**  
outbound\_bytes\_sent\_bottom

**Element type**  
watermark

*Table 760. 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\_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.

**Element identifier**  
outbound\_bytes\_received\_bottom

**Element type**  
watermark

*Table 761. 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.

## **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 762. 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 763. 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 764. 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 765. 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 766. 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\_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 767. 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 768. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Statement      |

Table 768. Snapshot Monitoring Information (continued)

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Application   | dcx_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\_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 769. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcx_dbase             | Statement      |
| DCS Application   | dcx_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 770. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| DCS Database      | dcx_dbase             | Statement      |
| DCS Application   | dcx_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 771. 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\_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 772. 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\_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 773. 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 774. 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\_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 775. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcs_dbase             | Statement      |



Table 775. Snapshot Monitoring Information (continued)

| Snapshot Level    | Logical Data Grouping | Monitor Switch |
|-------------------|-----------------------|----------------|
| 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\_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 776. 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 777. 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 778. 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.

**Element identifier**

max\_data\_received\_31999

**Element type**

counter

*Table 779. 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 780. 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\_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.

**Element identifier**  
max\_data\_received\_64000

**Element type**  
counter

*Table 781. 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\_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 782. 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 783. 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.)

**Element identifier**

max\_network\_time\_1\_ms

**Element type**

counter

Table 784. 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\_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 785. 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\_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 786. 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\_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.)

### Element identifier

max\_network\_time\_100\_ms

### Element type

counter

Table 787. 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\_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.)

### Element identifier

max\_network\_time\_500\_ms

### Element type

counter

Table 788. 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\_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.)

**Element identifier**

max\_network\_time\_gt500\_ms

**Element type**

counter

*Table 789. 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.

## 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 790. Snapshot Monitoring Information*

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dcс_dbase             | Statement, Timestamp |
| DCS Application   | dcс_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.

## 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 791. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dc_s_dbase            | Statement, Timestamp |
| DCS Application   | dc_s_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.

## xid - Transaction ID

A unique transaction identifier (across all databases) generated by a transaction manager in a two-phase commit transaction.

**Element identifier**

xid

**Element type**

information

Table 792. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| DCS Application | dc_s_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.

## 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. 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.

**Element identifier**

elapsed\_exec\_time

**Element type**

time

Table 793. 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 |



Table 793. Snapshot Monitoring Information (continued)

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| 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 `dcz_dbase`, `dcz_appl`, `dcz_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.

## 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.

### Element identifier

host\_response\_time

### Element type

time

Table 794. Snapshot Monitoring Information

| Snapshot Level    | Logical Data Grouping | Monitor Switch       |
|-------------------|-----------------------|----------------------|
| DCS Database      | dcz_dbase             | Statement            |
| DCS Application   | dcz_appl              | Statement, Timestamp |
| DCS Statement     | dcz_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):

$(\text{outbound bytes sent} + \text{outbound bytes received}) / \text{host response time}$

## 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 795. 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.)

**Element identifier**

num\_transmissions\_group

**Element type**

information

*Table 796. 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 |

## 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 797. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcс_dbase             | Timestamp      |

**Usage** Use this element as an indicator of the time it currently takes applications to connect to a particular host database.

## 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 798. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcс_dbase             | Timestamp      |

**Usage** Use this element as an indicator of the length of time that applications are maintaining connections to a host database.

## 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.

**Element identifier**  
gw\_comm\_errors

**Element type**  
counter

*Table 799. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcс_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\_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.

**Element identifier**

gw\_comm\_error\_time

**Element type**

timestamp

*Table 800. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| DCS Database   | dcс_dbase             | Timestamp      |

### Usage

Use this element for problem determination, in conjunction with Communication Error and the communication error logged in administration notification log.

## 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 801. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | stmt                  | Statement      |
| DCS Statement  | dcс_stmt              | Statement      |

*Table 802. Event Monitoring Information*

| Event Type             | Logical Data Grouping | Monitor Switch |
|------------------------|-----------------------|----------------|
| Deadlocks with Details | event_detailed_dlconn | -              |
| Statements             | event_stmt            | -              |

**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.

## Transaction processor monitoring monitor elements

In a transaction monitor or application server (multi-tier) environment, application users do not issue SQL requests directly. Instead, they request the transaction processor monitor (for example, CICS, TUXEDO, or ENCINA running on a UNIX or Windows server) or application server to execute a business transaction. Each business transaction is an application part that issues SQL requests to the database server. Because the SQL requests are issued by an intermediate server, the database server has no information about the original client that caused the execution of the SQL request.

Developers of transaction processor monitor (TP monitor) transactions or application server code can use the sqleseti - Set Client Information API to provide information about the original client to the database server. This information can be found in the following monitor elements.

### **tpmon\_client\_userid - TP Monitor Client User ID**

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 that applied to this activity.

**Element identifier**

tpmon\_client\_userid

**Element type**

information

*Table 803. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |
| Activities      | event_activity        | -              |

**Usage** Use this element in application server or TP monitor environments to identify the end-user for whom the transaction is being executed.

### **tpmon\_client\_wkstn - TP Monitor Client Workstation Name**

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 that applied to this activity.

**Element identifier**

tpmon\_client\_wkstn

**Element type**

information

*Table 804. Snapshot Monitoring Information*

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcs_appl              | Basic          |
| Activities      | event_activity        | -              |

**Usage** Use this element to identify the user's machine by node ID, terminal ID, or similar identifiers.

### **tpmon\_client\_app - TP Monitor Client Application Name**

Identifies the server transaction program performing the transaction, if the sqleseti API was issued in this connection. The current value of the client\_applname special register that applied to this activity.

**Element identifier**

tpmon\_client\_app

**Element type**

information

Table 805. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcx_appl              | Basic          |
| Activities      | event_activity        | -              |

**Usage** Use this element for problem determination and accounting purposes.

### **tpmon\_acc\_str - TP Monitor Client Accounting String**

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 that applied to this activity.

**Element identifier**

tpmon\_acc\_str

**Element type**

information

Table 806. Snapshot Monitoring Information

| Snapshot Level  | Logical Data Grouping | Monitor Switch |
|-----------------|-----------------------|----------------|
| Application     | appl_info             | Basic          |
| DCS Application | dcx_appl              | Basic          |
| Activities      | event_activity        | -              |

**Usage** Use this element for problem determination and accounting purposes.

---

## **Federated database systems monitor elements**

A federated system is a multidatabase server that provides remote data access. It provides client access to diverse data sources that can reside on different platforms, both IBM and other vendors, relational and non-relational. It integrates access to distributed data and presents a single database image of a heterogeneous environment to its users.

The following elements list 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.

### **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 807. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase_remote          | Basic          |

Table 807. Snapshot Monitoring Information (continued)

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_remote           | Basic          |

**Usage** Use this element to identify the data source whose access information has been collected and is being returned.

## 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.

**Element identifier**  
disconnects

**Element type**  
counter

Table 808. 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.

## 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.

**Element identifier**  
insert\_sql\_stmts

**Element type**  
counter

Table 809. 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\_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.

### Element identifier

update\_sql\_stmts

### Element type

counter

Table 810. 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\_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.

### Element identifier

delete\_sql\_stmts

### Element type

counter



Table 811. 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:

$$\text{write\_activity} = \frac{(\text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}{(\text{SELECT statements} + \text{INSERT statements} + \text{UPDATE statements} + \text{DELETE statements})}$$

## 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.

### Element identifier

create\_nickname

### Element type

counter

Table 812. 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.

## passthru - 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.

### Element identifier

passthru

**Element type**  
counter

*Table 813. 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 utilize native support.

## 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.

**Element identifier**  
stored\_procs

**Element type**  
counter

*Table 814. 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.

## 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.

**Element identifier**  
remote\_locks

**Element type**  
counter

*Table 815. 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.

## 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.

**Element identifier**

sp\_rows\_selected

**Element type**

counter

*Table 816. 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:

$$\text{average time} = \text{aggregate stored proc. response time} / \text{rows returned}$$

## 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.

**Element identifier**

select\_time

**Element type**

counter

Table 817. 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.

## 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.

### Element identifier

insert\_time

### Element type

counter

Table 818. 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.

## 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.

**Element identifier**

update\_time

**Element type**

counter

*Table 819. 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.

## 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.

**Element identifier**

delete\_time

**Element type**

counter

*Table 820. 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.

## 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.

### Element identifier

create\_nickname\_time

### Element type

counter

Table 821. 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.

## 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.

### Element identifier

passthru\_time

### Element type

counter

Table 822. 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.

## 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.

### Element identifier

stored\_proc\_time

### Element type

counter

Table 823. 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.

## 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.

### Element identifier

remote\_lock\_time

### Element type

counter

Table 824. 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.

---

## Workload management monitor elements

The following monitor elements provide information about activities, threshold violations, and workload management statistics.

### activate\_timestamp - Activate timestamp monitor element

The time when an event monitor was activated.

**Element identifier**

activate\_timestamp

**Element type**

timestamp

*Table 825. 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 above event types.

### activity\_collected - Activity collected monitor element

This element indicates whether or not activity event monitor records are to be collected for a violated threshold.

**Element identifier**

activity\_collected

**Element type**

information

*Table 826. 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. Used with **appl\_id** and **uow\_id** in an activities event monitor record, this monitor element uniquely identifies an activity that has been collected. Used



with **appl\_id** and **uow\_id** in a threshold violations event monitor record, this monitor element uniquely identifies an activity that has violated a threshold.

**Element identifier**  
activity\_id

**Element type**  
information

*Table 827. Event Monitoring Information*

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Activities           | event_activity            | -              |
| Activities           | event_activitystmt        | -              |
| Activities           | event_activityvals        | -              |
| Threshold violations | event_thresholdviolations | -              |

## 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 **uow\_id** and **agent\_id** monitor elements to uniquely identify an activity.

## 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.

**Element identifier**  
activity\_secondary\_id

**Element type**  
information

*Table 828. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |
| Activities | event_activitystmt    | -              |
| Activities | event_activityvals    | -              |

## 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\_type - Activity type monitor element

The type of the activity to which this activity record applies.

### Element identifier

activity\_type

### Element type

information

Table 829. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## Usage

The possible values are:

- LOAD
- READ\_DML
- WRITE\_DML
- DDL
- CALL
- OTHER

At remote partitions, the value of this monitor element is always OTHER.

## act\_exec\_time - Activity execution time monitor element

Time spent executing at this partition, 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 towards 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 towards 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.

### Element identifier

act\_exec\_time

### Element type

time

Table 830. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## Usage

This element can be used alone to know the elapsed time spent executing the activity by DB2 on each partition. This element can also be used together with

**time\_started** and **time\_completed** monitor elements on the coordinator partition 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\_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.

**Element identifier**  
act\_total

**Element type**  
counter

Table 831. 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.

## arm\_correlator - Application response measurement correlator monitor element

Identifier of a transaction in the Application Response Measurement (ARM) standard.

**Element identifier**  
arm\_correlator

**Element type**  
information

Table 832. 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.

## bin\_id - Histogram bin identifier monitor element

The identifier of a histogram bin. The **bin\_id** is unique within a histogram.

**Element identifier**  
bin\_id

**Element type**  
information

*Table 833. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

## Usage

Use this element to distinguish bins within the same histogram.

## 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.

**Element identifier**  
bottom

**Element type**  
information

*Table 834. 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.

## 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.

**Element identifier**  
concurrent\_act\_top

**Element type**  
watermark

*Table 835. 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 partition 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.

### Element identifier

concurrent\_connection\_top

### Element type

watermark

Table 836. 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.

### Element identifier

concurrent\_wlo\_act\_top

### Element type

watermark

Table 837. 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 partition 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.

### Element identifier

concurrent\_wlo\_top

### Element type

watermark

Table 838. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |

## Usage

Use this element to know the highest concurrency of workload occurrences reached on a partition for a workload in the time interval collected.

## 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.

### Element identifier

coord\_act\_aborted\_total

### Element type

counter

Table 839. 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.

### Element identifier

coord\_act\_completed\_total

### Element type

counter

Table 840. 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 partitions.

### **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.

#### **Element identifier**

coord\_act\_lifetime\_top

#### **Element type**

watermark

*Table 841. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wcstats         | -              |
| Statistics | event_scstats         | -              |

## 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\_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.

#### **Element identifier**

coord\_act\_rejected\_total

#### **Element type**

counter

*Table 842. 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\_partition\_num - Coordinator partition number monitor element

The partition number of the coordinator partition of this activity.

### Element identifier

coord\_partition\_num

### Element type

information

Table 843. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Activities           | event_activity            | -              |
| Threshold violations | event_thresholdviolations | -              |

## Usage

This element allows the coordinator partition to be identified for activities that have records on partitions other than the coordinator.

## 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.

### Element identifier

cost\_estimate\_top

### Element type

watermark

Table 844. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this element to determine the highest DML activity estimated cost reached on a partition for a service class or work class in the time interval collected.

## 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 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. Units are milliseconds.

**Element identifier**

coord\_act\_lifetime\_avg

**Element type**

information

*Table 845. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

**Usage**

Use this statistic to determine the arithmetic mean of the lifetime for coordinator activities associated with a service subclass 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\_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 elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class. Units are milliseconds. 0 associated with this service subclass that completed or aborted since the last reset. Returns -1 when COLLECT AGGREGATE ACTIVITY DATA of service class is NONE. Units are milliseconds.

**Element identifier**

coord\_act\_queue\_time\_avg

**Element type**

information

*Table 846. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this statistic to determine the arithmetic mean of the queue time for coordinator activities associated with a service subclass 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\_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. Units are milliseconds.

#### **Element identifier**

coord\_act\_exec\_time\_avg

#### **Element type**

information

*Table 847. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this statistic to determine the arithmetic mean of execution time for coordinator activities associated with a service subclass 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.

### **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.

**Element identifier**

request\_exec\_time\_avg

**Element type**

information

*Table 848. 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 database partition 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.

**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. Units are milliseconds.

**Element identifier**

coord\_act\_est\_cost\_avg

**Element type**

information

*Table 849. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

**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 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\_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. Units are milliseconds.

### **Element identifier**

coord\_act\_interarrival\_time\_avg

### **Element type**

information

*Table 850. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## **Usage**

Use this statistic to determine the arithmetic mean between arrivals of coordinator activities at nesting level 0 associated with this service subclass 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.

## **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.

### **Element identifier**

db\_work\_action\_set\_id

**Element type**  
information

*Table 851. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## 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.

**Element identifier**  
db\_work\_class\_id

**Element type**  
information

*Table 852. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## 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.

## histogram\_type - Histogram type monitor element

The type of the histogram, in string format.

There are six histogram types.

### CoordActQueueTime

A histogram the time non-nested activities spend queued (for example, in a threshold queue), measured on the coordinator partition.

### CoordActExecTime

A histogram of the time non-nested activities spend executing at the coordinator partition. 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.

### CoordActLifetime

A histogram of the elapsed lifetime of non-nested activities, measured on the coordinator partition from the time when an activity enters the system until the activity completes execution. Lifetime includes time the activity spends initializing, queued and executing.

### CoordActInterArrivalTime

A histogram of the time interval between the arrival of non-nested coordinator activities.

**CoordActEstCost**

A histogram of the estimated cost of non-nested DML activities.

**ReqExecTime**

A histogram of request execution times. Includes all requests on both coordinator and non-coordinator partitions including those requests not associated with an activity.

**Element identifier**

histogram\_type

**Element type**

information

*Table 853. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |

**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.

**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.

**Element identifier**

last\_wlm\_reset

**Element type**

information

*Table 854. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| 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**.

**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.

**Element identifier**

num\_threshold\_violations

**Element type**  
counter

*Table 855. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Basic          |

For snapshot monitoring, this counter can be reset.

*Table 856. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## 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.

## 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.

**Element identifier**  
number\_in\_bin

**Element type**  
information

*Table 857. 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.

## 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.

**Element identifier**  
parent\_activity\_id

**Element type**  
information

*Table 858. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## Usage

Use this element along with the **parent\_uow\_id** element and **appl\_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.

**Element identifier**

parent\_uow\_id

**Element type**

information

*Table 859. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

### 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.

## prep\_time - Preparation time monitor element

Time in milliseconds required to prepare an SQL statement if the activity is an SQL statement.

**Element identifier**

prep\_time

**Element type**

time

*Table 860. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

### Usage

This element can be used to identify how much of the activity's total lifetime was spent preparing the SQL statement, if this was an SQL activity.

## queue\_assignments\_total - Queue assignments total monitor element

The number of connections or activities that were assigned to this threshold queue since the last reset.

**Element identifier**

queue\_assignments\_total

**Element type**

counter

*Table 861. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_qstats          | -              |



## Usage

This element can be used to determine the number of activities or connections that were 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\_size\_top - Queue size top monitor element

Highest queue size that has been reached since the last reset.

**Element identifier**

queue\_size\_top

**Element type**

watermark

*Table 862. 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.

**Element identifier**

queue\_time\_total

**Element type**

counter

*Table 863. 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.

### 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 database partition for which this information is recorded. On DPF systems, these values may not reflect the correct totals for the whole activity.

**Element identifier**

rows\_fetched

**Element type**  
counter

*Table 864. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | Statement      |

## Usage

See the **rows\_read** monitor element for details.

## 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.

**Note:** This monitor element reports only the values for the database partition for which this record is recorded. On DPF systems, these values may not reflect the correct totals for the whole activity.

**Element identifier**  
rows\_modified

**Element type**  
counter

*Table 865. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | Statement      |

## Usage

See the **rows\_written** monitor element for details.

## 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.

**Element identifier**  
rows\_returned

**Element type**  
counter

*Table 866. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## 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.

#### **Element identifier**

rows\_returned\_top

#### **Element type**

watermark

*Table 867. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this element to know the highest DML activity actual rows returned reached on a partition for a service class or work class in the time interval 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.

#### **Element identifier**

sc\_work\_action\_set\_id

#### **Element type**

information

*Table 868. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## 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.

### Element identifier

sc\_work\_class\_id

### Element type

information

Table 869. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## 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.

## section\_env - Section environment monitor element

A handle that gives details of an activity's section.

### Element identifier

section\_env

### Element type

information

Table 870. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activitystmt    | -              |

## Usage

This element is to be used with future IBM tools for extracting section information for the activity described in this record

## service\_class\_id - Service class ID monitor element

Unique ID of the service class. Can be used for doing joins with the histogrambins table.

### Element identifier

service\_class\_id

### Element type

information

Table 871. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |
| Statistics | event_scstats         | -              |

## Usage

Use this element with the **statistics\_timestamp** and **partition\_number** monitor elements to link histogram bin records with service class statistics records.

### **service\_subclass\_name** - Service subclass name monitor element

The name of the service subclass to which this activity record or statistics record applies.

#### Element identifier

service\_subclass\_name

#### Element type

information

Table 872. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |
| Statistics | event_scstats         | -              |
| Statistics | event_qstats          | -              |

## 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 the service superclass to which this activity record or statistics record applies.

#### Element identifier

service\_superclass\_name

#### Element type

information

Table 873. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |
| Statistics | event_scstats         | -              |
| Statistics | event_qstats          | -              |

## 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.

## statistics\_timestamp - Statistics timestamp monitor element

The time at which this statistics record was generated.

### Element identifier

statistics\_timestamp

### Element type

information

Table 874. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wlstats         | -              |
| 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.

## temp\_tablespace\_top - Temporary table space top monitor element

The high watermark 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.

### Element identifier

temp\_tablespace\_top

### Element type

watermark

Table 875. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_scstats         | -              |
| Statistics | event_wcstats         | -              |

## Usage

Use this element to determine the highest DML activity system temporary table space usage reached on a partition for a service class 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. If aggregate activity data collection is not enabled for the service class or work class, a value of -1 is returned.

## threshold\_action - Threshold action monitor element

The action of the threshold to which this threshold violation record applies. Possible values include Stop and Continue.

**Element identifier**  
threshold\_action

**Element type**  
information

Table 876. 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 or was allowed to continue executing. If the activity was stopped, the application that submitted the activity will have received an SQL4712N error.

## 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

**Element identifier**  
threshold\_domain

**Element type**  
information

Table 877. 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.

### Element identifier

threshold\_maxvalue

### Element type

information

Table 878. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Threshold violations | event_thresholdviolations | -              |

## 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.

## threshold\_name - Threshold name monitor element

The unique name of the threshold responsible for this queue.

### Element identifier

threshold\_name

### Element type

information

Table 879. 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.

### Element identifier

threshold\_predicate

### Element type

information



Table 880. 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 statistics or threshold violation monitor elements for analysis of a threshold violation.

## 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.

### Element identifier

threshold\_queuesize

### Element type

information

Table 881. 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.

### Element identifier

thresholdid

### Element type

information

Table 882. 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.

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 or if the activity was captured while it was in progress.

### Element identifier

time\_completed

### Element type

information

Table 883. 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\_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.

### Element identifier

time\_created

### Element type

information

Table 884. 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.

### Element identifier

time\_of\_violation

### Element type

information

Table 885. 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\_started - Time started monitor element

The time at which the activity described by this activity record began executing. This element is a local timestamp.

**Element identifier**  
time\_started

**Element type**  
information

Table 886. 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.

## 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.

**Element identifier**  
top

**Element type**  
information

Table 887. 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.

## uow\_id - Unit of work ID monitor element

The unit of work ID to which this activity record applies. The unit of work ID is unique within an application handle.

**Element identifier**  
uow\_id

**Element type**  
information

Table 888. Event Monitoring Information

| Event Type           | Logical Data Grouping     | Monitor Switch |
|----------------------|---------------------------|----------------|
| Activities           | event_activity            | -              |
| Activities           | event_activitystmt        | -              |
| Activities           | event_activityvals        | -              |
| Threshold violations | event_thresholdviolations | -              |

## 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.

## wlo\_completed\_total - Workload occurrences completed total monitor element

The number of workload occurrences to complete since last reset.

### Element identifier

wlo\_completed\_total

### Element type

counter

Table 889. 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.

### Element identifier

work\_action\_set\_id

### Element type

information

Table 890. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_histogrambin    | -              |
| Statistics | event_wcstats         | -              |

## 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.

### **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.

#### **Element identifier**

work\_action\_set\_name

#### **Element type**

information

*Table 891. 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.

#### **Element identifier**

work\_class\_id

#### **Element type**

information

*Table 892. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wcstats         | -              |
| Statistics | event_histogrambin    | -              |

## Usage

Use this element in conjunction with other statistics elements for analysis of 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.

#### **Element identifier**

work\_class\_name

**Element type**  
information

*Table 893. 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

The ID of the workload to which this activity, application, or workload statistics record belongs.

**Element identifier**  
workload\_id

**Element type**  
information

*Table 894. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Application    | appl_info             | Basic          |

*Table 895. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |
| Activities | event_activity        | -              |

## Usage

Use this ID to uniquely identify the workload to which this activity, application, or workload statistics record belongs.

## workload\_name - Workload name monitor element

Name of the workload to which this statistics record applies.

**Element identifier**  
workload\_name

**Element type**  
information

*Table 896. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Statistics | event_wlstats         | -              |

## Usage

Use this element in conjunction with other statistics elements for analysis of a workload.

### **workload\_occurrence\_id - Workload occurrence identifier monitor element**

The ID of the workload occurrence to which this activity belongs.

#### Element identifier

workload\_occurrence\_id

#### Element type

*Table 897. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Activities | event_activity        | -              |

## Usage

Use this to identify the workload occurrence that submitted the activity.

---

## Real-time statistics monitor elements

The following monitor elements provide information about real-time statistics gathering.

### **stats\_cache\_size – Size of statistics cache monitor element**

The current size of the statistics cache, 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.

#### Element identifier

stats\_cache\_size

#### Element type

Gauge

*Table 898. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | -              |

*Table 899. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## 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\_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.

### Element identifier

stats\_fabrications

### Element type

Counter

*Table 900. Snapshot Monitoring Information*

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

*Table 901. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## 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.

## **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.

### Element identifier

sync\_runstats

### Element type

Counter



Table 902. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

Table 903. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## 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.

## 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.

### Element identifier

async\_runstats

### Element type

Counter

Table 904. Snapshot Monitoring Information

| Snapshot Level | Logical Data Grouping | Monitor Switch |
|----------------|-----------------------|----------------|
| Database       | dbase                 | Statement      |

For snapshot monitoring, this counter can be reset.

Table 905. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |

## 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.

## 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.

### Element identifier

stats\_fabricate\_time

### Element type

Time

Table 906. 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 907. Event Monitoring Information

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Statement  | event_stmt            | -              |

## 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.

## 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.

### Element identifier

sync\_runstats\_time

**Element type**  
time

*Table 908. 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 909. Event Monitoring Information*

| Event Type | Logical Data Grouping | Monitor Switch |
|------------|-----------------------|----------------|
| Database   | event_db              | -              |
| Statement  | event_stmt            | -              |

## 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.



## Chapter 10. 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                           |
|--------------------------------------------------------------------------|---------------------------------------|
| Analyzing event monitor output with a GUI tool                           | db2eva                                |
| 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_V95 table function |

| Monitoring task                                                           | SQL 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_V95 table function                   |
| Getting an application level snapshot                                     | SNAPAPPL administrative view and SNAP_GET_APPL_V95 table function               |
| Getting an application level snapshot                                     | SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO_V95 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_V95 table function                   |
| Getting a table space level snapshot                                      | SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function               |
| Getting a table space level snapshot for configuration information        | SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function     |
| Getting a table space level snapshot for container information            | SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 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_V91 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_V95 table function         |

---

## **Part 3. Monitoring database health**





---

## Chapter 11. 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 also has the capability to 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.

The health monitor checks the state of your system using health indicators to determine if 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 e-mail 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 very 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.

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 Health Center, 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 910. 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.

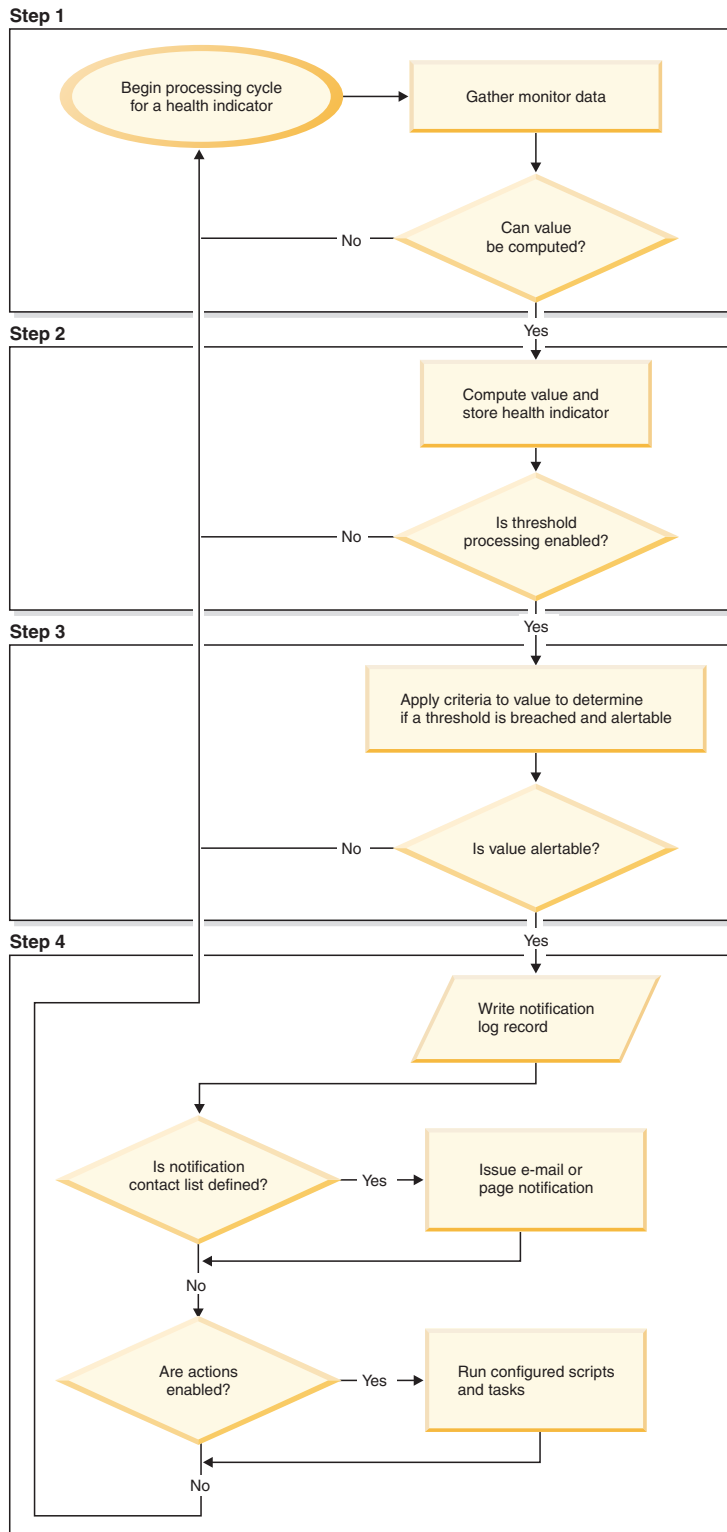


Figure 6. Health indicator process cycle

**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.
2. When migrating a Version 7 installation of the DB2 database on Windows, the value of the NOTIFYLEVEL database manager configuration parameter is not updated.

---

## Enabling health alert notification

To enable e-mail or pager notification when an alert is generated, you must set configuration parameters and specify contact information.

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.

To enable health alert notification:

1. Specify the SMTP\_SERVER parameter. The DAS configuration parameter, SMTP\_SERVER, specifies the location of the mail server to use when sending both e-mail 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 e-mail 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, C APIs, or the Health Center:

- 

**To specify contacts using the CLP:**

To define an e-mail 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);
}
...

```

### To specify contacts using the Health Center:

- a. Right-click the instance for which you want to define the health notification list.
- b. Click **Configure**, then click **Alert Notification**. The Configure Health Alert Notification window opens.
- c. If contacts do not appear in the left side of the window in the **Available** list, click **Manage Contacts**. The Contacts window opens with the system name preselected.
- d. Click **Add Contact**. The Add Contact window opens.
- e. Define a contact by supplying a name and an e-mail address. Select **Address is for a pager** if the specified e-mail address is for a pager.
- f. Click **OK**.
- g. Close the Contacts window and return to the Configure Health Alert Notification window. The new contact now appears in the **Contacts available** list.

- h. Move the contact to the **Health notification contact list** by clicking the right arrow button.
- i. Click **OK** to include the contact in the health notification list.

**Recommendation**

If you are experiencing difficulties with notification, select **Troubleshoot** below the Health notification contact list.

The Troubleshoot Health Alert Notification wizard opens.

---

## Chapter 12. Health Center overview

Use the Health Center to analyze and improve the health of DB2.

The following are examples of conditions that define what makes DB2 healthy:

- There are sufficient resources, such as free memory, table space containers, or logging storage, to accomplish tasks.
- Resources are used efficiently.
- Tasks complete within acceptable periods of time or without significant degradations in performance.
- Resources or database objects are not left indefinitely in unusable states.

From the Health Center you can also open other centers and tools to help you investigate and maintain the health of your database.

To open the Health Center on Intel platforms, from the **Start** menu, click **Start** → **Programs** → **IBM DB2** → **Monitoring Tools** → **Health Center**.

To open the Health Center using the command line on Intel platforms, run the following command:

```
db2hc
```

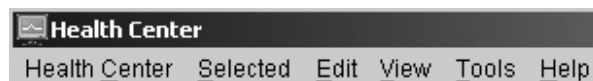
The following list categorizes some of the key tasks that you can do with the Health Center:

- “Enabling health alert notification” on page 514
  - Specifying contact settings and notification configuration parameters
  - Troubleshooting health alert notification
- “Configuring health indicators using the Health Center” on page 543
  - Enabling and disabling health indicator evaluation
  - Changing alert threshold and sensitivity settings
  - Running tasks and scripts when an alert occurs
- “Resolving health monitor alerts using the Health Center” on page 536
  - Using the Recommendation advisor to select and implement recommendations

### The Health Center interface

The Health Center interface has the following elements that help you determine and resolve problems related to the overall health of your system.

#### Health Center Menu bar



Use the menu bar to work with objects in the Health Center, open other administration centers and tools, and access online help.

The Health Center menu bar contains the following menus:

## Health Center toolbar



Use the toolbar icons below the menu bar to access other centers and tools, and to refresh the content view of the Health Center.

## Toggle buttons



Use the toggle buttons to select the alert states that appear in the Navigation view. Each button corresponds to a minimum alert severity that a database object needs in order to appear in the view. Selecting a different button only affects the display and does not affect the object itself.



Shows objects in alarm state



Shows objects in alarm and warning states



Shows objects in any alert state: alarm, warning, attention, normal, and not monitored.



Shows all objects

## Navigation view



Use the navigation view to display and work with instance and database objects. When you select an object in the Navigation view, the current alerts for that object and all its children are displayed in the Alerts view. To change the level of alert that object must have before the navigation view displays it, right click in the navigation view away from the listed objects. This will open a pop-up menu of alert levels. Select the alert levels that you want displayed. You can also choose what alert levels to display by clicking the toggle buttons.

## Alerts view

Use the Alerts view to display and work with current alerts. The Alerts view displays those alerts that currently exist for the object and its children database objects selected in the navigation view. For example, if you select an instance, alerts display for the instance and all its databases and table spaces. If you select a database, alerts display for the database and all table spaces for the database. Select and right-click one or more alerts in the Alerts view to invoke actions for them.

## Alerts view toolbar



Use the toolbar below the Alerts view to tailor the view of alerts in the Alerts view to suit your needs.



---

## Investigating alert conditions



---

## Chapter 13. Health monitor

The health monitor captures information about the database manager, database, table space and table space containers. 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 ten 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 has been 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 has not been 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.

To start or stop the health monitor from the Health Center, right-click an instance in the Health Center navigational view and select Start Health Monitor or Stop Health Monitor

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.

If you migrate back to DB2 UDB Version 8.1 from a later version of the DB2 database system, any registry changes that have been made are lost. The registry reverts to the version 8.1 HealthRules.reg file that contains the settings that existed before you upgraded and started using the settings in the newer registry file.

---

## 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.

To capture a database health snapshots using SQL table functions:

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.

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.

To capture a database health snapshot using the CLP

1. From the CLP, issue the GET HEALTH SNAPSHOT command with the desired 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:

## 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.

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.

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);

```

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:

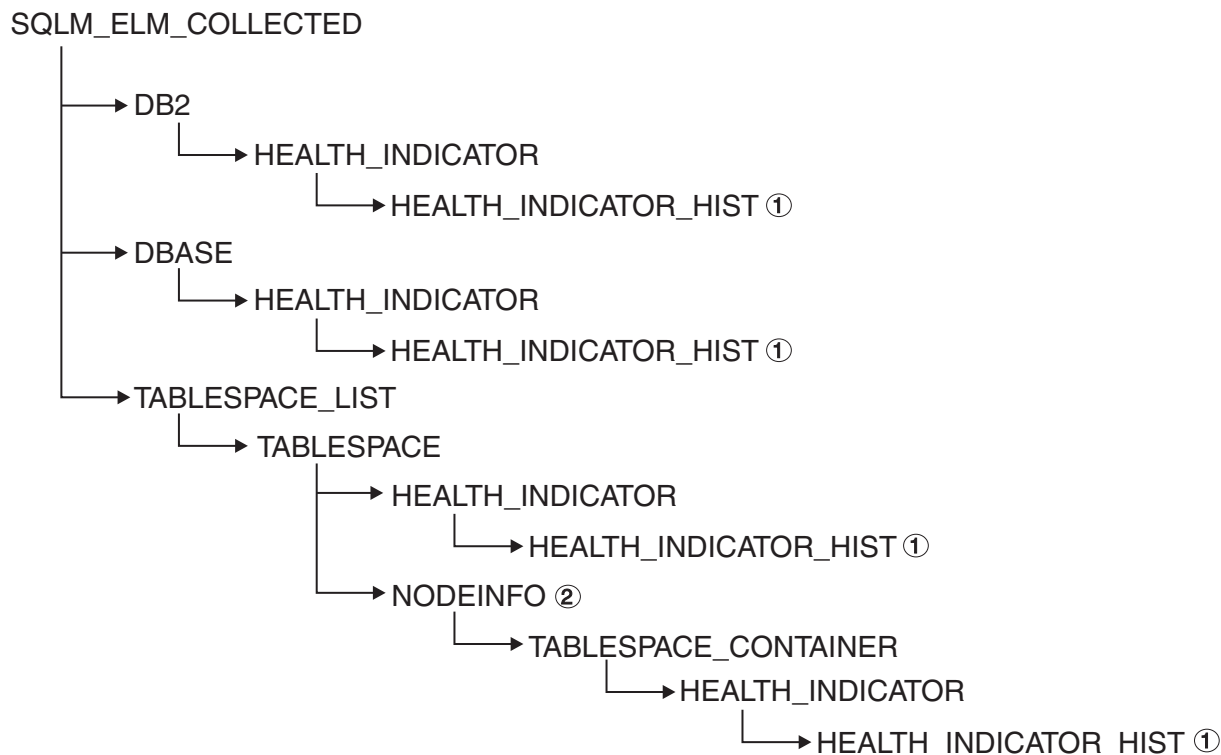


Figure 7. Health snapshot self-describing data stream

**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 911. 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>                                                                                                     |

Table 911. Aggregation of health indicator value, timestamp, and formula data (continued)

| Health indicator                                                                                                                                                       | Aggregation details                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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> | 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. |

**Note:** When taking a global snapshot on a single partition object, the output includes all the attributes because there are no partitions to aggregate.

---

## Graphical tools for the health monitor

### Health Center

The Health Center is a graphical administration tool designed to support management-by-exception. For all Windows, Linux, and UNIX instances and databases cataloged on the client, the Health Center provides:

- A central location to view the rolled up alert state of all instances and their databases
- A graphical interface to view current alerts on the instances and databases and their children objects
- A graphical interface to access details and recommended resolution actions for current alerts

To start the Health Center from the command line, type the db2hc command.

On Windows, you can also start the Health Center from the Start Menu by clicking **Start** → **Programs** → **IBM DB2** → <DB2 copy name> → **Monitoring Tools** → **Health Center**.

The Health Center has a navigation tree in the left panel and an Alerts view in the right panel. The contents of the navigation view are filtered based on the toggle button selected at the top of the navigation view.

The Health Center opens with the **Object in Any Alert State** toggle button selected, which helps to identify those instances with current alerts that should be addressed. When the **All Objects** toggle button is selected, all Windows, Linux, and UNIX instances cataloged on the client and their respective states are displayed. Instances without an icon do not have the health monitor running or are instances prior to version 8, which lack support for health monitor functionality.

When you select an instance, the Health Center requests status from the health monitor for the selected instance. The Alerts view fills with all current alerts for the instance, any of its databases, and any of the table spaces and table space containers of each database. If you expand the instance in the navigation view and select a child database object, the Alerts view is restricted to alerts for the selected database and any of its table spaces or table space containers.

The refresh icon is located in the upper-right corner of the Health Center. Clicking the refresh icon for immediate refresh, or setting a particular refresh interval, causes the Health Center to query the health monitor on the server for its current status. This query does not cause the health monitor to refresh the health indicator

evaluations. Each health indicator has a defined refresh interval. Only when the refresh interval has passed will the health indicator be reevaluated for alert state. Only the current status of the health indicators is shown on each timed refresh or requested refresh of the Health Center.

The Alerts view has a function to define customized views with specific customized columns and sorting orders. There are six predefined views in the Health Center that you can customize to your personal naming and categorization scheme. You can select the predefined views by using the toolbar at the bottom of the window or by selecting **Saved Views** in the **View** menu. To define your own customized views, click the **View** button on the toolbar at the bottom of the window, or use the **View** menu. The view that is selected for displaying data in the Alerts view is remembered on the next invocation of the Health Center.

To get the details for an alert, select the alert row in the Alerts view. Using the **Selected** menu, or by right-clicking the row, select **Show Details**. The Details window shows the detailed information for the alert including the object and partition where the alert occurred, the formula (if applicable), and value for the health indicator.

For threshold-based health indicators, the thresholds that were used in determining the alert condition are displayed. The Details window also displays additional information for the health indicator. This information might include values for configuration parameters or other monitor data that provides context for the alert. A description of the health indicator is displayed, including the purpose for the health indicator and why it is an important attribute to measure.

For collection state-based health indicators, the list of collection objects is displayed in the Objects in **Health Indicator Alert State** table. Object name, state, timestamp, and details are provided in the table.

A View History button is provided on the details page. History records are stored for the health indicator starting with the second refresh of the health indicator evaluation. Content is displayed in the View History dialog in the Health Center only after the history records are stored. The history of collection objects, for collection state-based health indicators, can be viewed by clicking the **View Collection History** button in the History window.

## Health Center Status Beacon

The Health Center Status Beacon is a visual indicator that can be enabled in the DB2 administration tools. When the Health Center is not open, the beacon will notify you of current alerts while you are working with other DB2 administration tools. The beacon is intended to prompt the user to open the Health Center because of an alert condition.

The Health Center Status Beacon has two different notification methods. One notification method uses a pop-up message. Another notification method uses a graphical beacon that displays on the right portion of the status line of open windows. The graphical beacon includes a button that provides single-click access to the Health Center.

Both beacon notification methods are enabled through the Tools Settings dialog. The "notify through pop-up" method controls the pop-up message notification, and the "notify through status line" method controls the visual beacon.

---

## 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 `sqllib/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 CLP

Recommendations can be retrieved using the `GET RECOMMENDATIONS` command from the CLP. The command syntax supports querying recommendations to resolve a specific health alert, such as a health indicator that has currently entered an alert state on a particular object.

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.

The command syntax also supports retrieval of the complete set of recommendations for a given 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 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. On the other hand, if recommendations are queried by health indicator name only, as in the first example below, the total set of possible recommendations will always be returned. In such cases, the CLP command is simply providing information about actions that a user should consider undertaking if they see an alert.

Retrieve the recommendations using the GET RECOMMENDATIONS command:

1. 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.

Take one of the following actions:

Launch DB2 tool: Utility Status Manager

The Utility Status Manager allows you to monitor the progress and change the priority of currently running utilities.

To open the Utility Status Manager:

1. From the Control Center, expand the object tree until you find the database that you want.
2. Right-click the database, and click Manage Utilities from the pop-up menu. The Utility Status Manager opens.

To view progress of the rollforward utility, right-click on the rollforward utility and select View Progress Details.

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.

Take one of the following actions:

Launch DB2 tool: Control Center Unquiesce Database

The Control Center has an option on a database that can be used to unquiesce the database.

To unquiesce a database:

1. From the Control Center, expand the object tree until you find the database that you want.
2. Right-click the database, and click Unquiesce from the pop-up menu. The database is unquiesced.

From the Command Line Processor, issue the commands shown in the following example:

```
CONNECT TO DATABASE database-alias
UNQUIESCE DATABASE
```

2. Suppose you observe that the health indicator **db.db\_heap\_util** has 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 has 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%).

Take one of the following actions:

Execute the following scripts 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;
```

Launch DB2 tool: Database Configuration Window

The Database Configuration window can be used to view and update database configuration parameters.

To open the Database Configuration window:

1. From the Control Center, expand the object tree until you find the databases folder.
2. Click the databases folder. Any existing database are displayed in the contents pane on the right side of the window.
3. Right-click the database that you want in the contents pane, and click Configure Parameters in the pop-up menu. The Database Configuration window opens.

On the Performance tab, update the database heap size parameter as suggested and click OK to apply the update.

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.

Take one of the following actions:

Launch DB2 tool: Memory Visualizer

The Memory Visualizer is used to monitor memory allocation within a DB2 instance. It can be used to monitor overall memory usage, and to update configuration parameters for individual memory components.

To open the Memory Visualizer:

1. From the Control Center, expand the object tree until you find the instances folder.
2. Click the instances folder. Any existing instances are displayed in the contents pane on the right side of the window.
3. Right-click the instance that you want in the contents pane, and click View Memory Usage in the pop-up menu. The Memory Visualizer opens.

To start the Memory Visualizer from the command line issue the db2memvis command.

The Memory Visualizer displays a hierarchical list of memory pools for the database manager. Database Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.

Click the check box on the Show Plot column for the Database Heap row to add the element to the plot.

3. For partitioned database systems you can query recommendations for a health indicator that has 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.

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.

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:

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 sqllib\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 ) ;`

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.

---

## Resolving health monitor alerts using the Health Center

The Health Center provides support to retrieve and implement recommended actions for alert conditions.

To resolve health monitor alerts using the Health Center:

1. From the Health Center alerts view, right-click the row of the alert that you want to resolve and select Recommendation Advisor from the pop-up menu. The Recommendation Advisor opens and displays the details of the alert in a format similar to the Details window.
2. Follow the steps of the Recommendation Advisor to select the most appropriate recommendation. The Recommendation Advisor provides the functionality to implement the recommendation.

There are two types of recommendations: investigation and recommendation. The following four types of actions are supported in the Recommendation Advisor for these recommendation types:

### Launching a graphical administration tool

This option will launch a graphical tool that will resolve or investigate the alert condition. The tool is launched in the context of the object against which the alert occurred.

### Updating configuration parameters

The configuration parameters requiring updates are listed with current and suggested values. The suggested value can be updated as needed.

### Running a DB2 command script

The recommendation action might require more than a single command. DB2 command scripts allow for multiple commands to be run to resolve the alert condition. For example, the Reorganization Required health indicator provides a DB2 command script action to run the utility.

### **Implementing an alternative resolution**

If the action cannot be accomplished within the DB2 administration toolset, instructions are provided to resolve the alert condition using alternate methods.

---

## **Health indicator configuration**

A default health monitor configuration is provided during installation. This 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.

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 will configured alert actions be 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 in the Task Center 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:\sqllib\DB2`.
- On UNIX, HealthRules.reg is stored in `~/<SQLLIB_PATH>/cfg`. For example, `~/home/sqllib/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.

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 parameter names for these settings are, respectively:

- 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.

- 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.

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.

Combinations of the `objType` and `defaultType` parameters in the `db2GetAlertCfgData` structure allow access to the various levels of health indicator configuration.

Table 912. Settings for `objType` and `defaultType` to access configuration levels

| Setting          | <code>objType</code> and <code>defaultType</code>                                                                                                                                                                                                                                                       |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Factory settings | <code>objType = DB2ALERTCFG_OBJTYPE_{DBM   DATABASES   TABLESPACES   CONTAINERS}</code> and <code>defaultType = DB2ALERTCFG_DEFAULT</code>                                                                                                                                                              |
| Global settings  | <code>objType = DB2ALERTCFG_OBJTYPE_{DBM   DATABASES   TABLESPACES   CONTAINERS}</code><br>and <code>defaultType = DB2ALERTCFG_NOT_DEFAULT</code><br><br>or<br><br><code>objType = DB2ALERTCFG_OBJTYPE_{DATABASE   TABLESPACE   CONTAINER}</code> and<br><code>defaultType = DB2ALERTCFG_DEFAULT</code> |
| Object settings  | <code>objType = DB2ALERTCFG_OBJTYPE_{DATABASE   TABLESPACE   CONTAINER}</code> and <code>defaultType = DB2ALERTCFG_NOT_DEFAULT</code>                                                                                                                                                                   |

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 `sqllib\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 sqllib\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);
```

## Configuring health indicators using the Health Center

The Health Center provides graphical interfaces to view, update, and reset health indicator configurations. The configuration for health indicators is stored in the health monitor within the instance.

To define, change, enable or disable threshold or sensitivity settings for a health indicator, and to define, change, enable or disable running tasks or scripts when a health alert occurs for a health indicator, you must have one of the following authorities:

- SYSADM
- SYSMANT
- SYSCTRL

You can adjust the health indicator settings for an instance, the global health indicator settings for database objects contained in the instance, and for individual database objects.

1. To configure health indicators using the Health Center:
  - a. Select the instance whose health indicators you want to configure.
  - b. From the **Selected** menu, or from the right-click menu, click **Configure** then **Health Indicator Settings**. The Health Indicator Configuration Launchpad opens.
  - c. Each level of configuration settings that can be updated has a button on the launchpad. Select the button for the level of configuration that you want to view, update, or reset. Each button launches a Health Indicator Configuration window at the chosen level of configuration settings.
  - d. To update the health indicator settings, select the row of the health indicator in the Current health indicator settings table.
  - e. From the **Select** menu, or from the right-click menu, select **Edit**. The Configure Health Indicator notebook opens and displays the following information:
    - A description of the health indicator is provided by clicking **Tell Me More**.
    - The evaluation of the health indicator can be enabled and disabled using the **Evaluate** check box.

**Note:** The **Evaluate** flag can also be disabled from the Health Center Alerts View for current alerts through the right-click menu option on a current alert. This option will disable the evaluation of the health indicator on the next refresh of the indicator in the health monitor. When selecting **Disable evaluation** for an alert in the Health Center, the evaluation flag is set to false for the health indicator, but the alert will not be removed from the Alerts view until the following events take place:

- The health monitor refresh interval for that particular health indicator is reached
  - The health monitor refreshes the health indicator evaluation
  - The Health Center refreshes its view of status
- On the Alert page, for threshold-based health indicators, the warning and alarm thresholds can be updated. The sensitivity for any health indicator can also be set on this page.

- On the Actions page, a task or script action can be selected to run when an alert occurs. Actions can be configured to run on warning or alarm conditions for threshold-based health indicators or on any non-normal condition for state-based health indicators. You can enable or disable the execution of actions by selecting or deselecting the **Enable actions** check box. To add, update, or remove task or script actions, use the buttons beside the **Script actions** and **Task actions** tables.
2. To view the factory health indicator settings for the instance:
    - a. In the Health Indicator Configuration launchpad, click **Instance Settings**.
    - b. In the Instance Health Indicator Configuration window, click **View Default**.
  3. To view the global health indicator settings for databases, table spaces, or table space containers:
    - a. In the Health Indicator Configuration launchpad, click **Global Settings**.
    - b. Select the object type in the Global Health Indicator Configuration window.
    - c. To view the factory defaults for these global settings, click **View Default**
  4. To view the health indicator settings for a database object:
    - a. In the Health Indicator Configuration launchpad, click **Object Settings**.
    - b. Select the object in the Object Health Indicator Configuration window.
    - c. To view the global default health indicator settings for this object type, click **View Default**.

In each of these windows, to reset the settings of all the displayed health indicators to their defaults, click **Reset to Default**. You can also reset individual health indicators by right-clicking one or more health indicators that you want in the **Current health indicator settings** field and selecting **Reset to Default** from the pop-up menu.

## 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 passwd001')
```

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')

```



---

## Part 4. 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.

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 Health Center, 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 913. Health indicator conditions at different refresh intervals*

| <b>Refresh interval</b> | <b>Value of ts.ts_util (Table space utilization) health indicator</b> | <b>State of ts.ts_util health indicator</b> | <b>Health monitor response</b>                                                 |
|-------------------------|-----------------------------------------------------------------------|---------------------------------------------|--------------------------------------------------------------------------------|
| 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          |

## Chapter 14. Health monitor interface mappings to logical data groups

The following table lists all the supported health snapshot request types.

Table 914. 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                         |
|                                                                | get health snapshot for dbm<br>show detail                                               | HEALTH_DBM_HI_HIS  | health_indicator_history                 |
| SQLMA_DBASE                                                    | get health snapshot for<br>database on <i>dbname</i>                                     | HEALTH_DB_INFO     | dbase                                    |
|                                                                |                                                                                          | HEALTH_DB_HI       | health_indicator                         |
|                                                                | get health snapshot for<br>database on <i>dbname</i> show<br>detail                      | HEALTH_DB_HI_HIS   | health_indicator_history                 |
| SQLMA_DBASE with<br>SQLM_HMON_OPT_COLL_FULL in<br>the agent_id | get health snapshot for<br>database on <i>dbname</i> with full<br>collection             | HEALTH_DB_HIC      | health_indicator, hi_obj_list            |
|                                                                | get health snapshot for<br>database on <i>dbname</i> show<br>detail with full collection | HEALTH_DB_HIC_HIST | health_indicator_history,<br>hi_obj_list |
| SQLMA_DBASE_ALL                                                | get health snapshot for all<br>databases                                                 | HEALTH_DB_INFO     | dbase                                    |
|                                                                |                                                                                          | HEALTH_DB_HI       | health_indicator                         |
|                                                                | get health snapshot for all<br>databases show detail                                     | HEALTH_DB_HI_HIS   | health_indicator_history                 |
| SQLMA_DBASE_TABLESPACES                                        | get health snapshot for<br>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<br>tablespaces on <i>dbname</i> show<br>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.

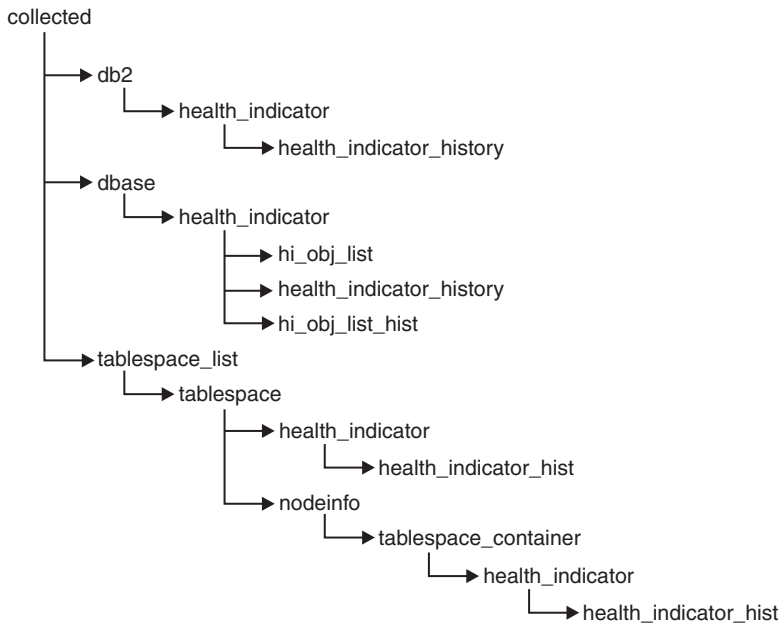


Figure 8. Health snapshot logical data groupings



## Chapter 15. Health indicators summary

The following tables list all health indicators, grouped by category.

*Table 915. 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 554 |

*Table 916. 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 555    |
| Automatic Resize Table Space Utilization | ts.ts_util_auto_resize   | "ts.ts_util_auto_resize - Automatic resize table space utilization health indicator" on page 556 |
| Table Space Utilization                  | ts.ts_util               | "ts.ts_util - Table Space Utilization" on page 556                                               |
| Table Space Container Utilization        | tsc.tscont_util          | "tsc.tscont_util - Table Space Container Utilization" on page 557                                |
| Table Space Operational State            | ts.ts_op_status          | "ts.ts_op_status - Table Space Operational State" on page 558                                    |
| Table Space Container Operational State  | tsc.tscont_op_status     | "tsc.tscont_op_status - Table Space Container Operational State" on page 558                     |
| Table Space Automatic Resize Status      | ts.ts_auto_resize_status | "ts.ts_auto_resize_status - Table space automatic resize status health indicator" on page 555    |

*Table 917. 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 558            |
| Shared Sort Memory Utilization           | db.sort_shrmem_util     | "db.sort_shrmem_util - Shared Sort Memory Utilization" on page 559               |
| Percentage of Sorts That Overflowed      | db.spilled_sorts        | "db.spilled_sorts - Percentage of Sorts That Overflowed" on page 560             |
| Long Term Shared Sort Memory Utilization | db.max_sort_shrmem_util | "db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization" on page 560 |

*Table 918. Database manager health indicators*

| Name                       | Identifier        | Additional Information                                       |
|----------------------------|-------------------|--------------------------------------------------------------|
| Instance Operational State | db2.db2_op_status | "db2.db2_op_status - Instance Operational State" on page 561 |

Table 918. Database manager health indicators (continued)

| Name                                  | Identifier | Additional Information                              |
|---------------------------------------|------------|-----------------------------------------------------|
| Instance Highest Severity Alert State | –          | “Instance Highest Severity Alert State” on page 561 |

Table 919. Database health indicators

| Name                                  | Identifier      | Additional Information                                     |
|---------------------------------------|-----------------|------------------------------------------------------------|
| Database Operational State            | db.db_op_status | “db.db_op_status - Database Operational State” on page 562 |
| Database Highest Severity Alert State | –               | “Database Highest Severity Alert State” on page 562        |

Table 920. Maintenance health indicators

| Name                                            | Identifier         | Additional Information                                            |
|-------------------------------------------------|--------------------|-------------------------------------------------------------------|
| Reorganization Required                         | db.tb_reorg_req    | “db.tb_reorg_req - Reorganization Required” on page 563           |
| Statistics Collection Required health indicator | db.tb_runstats_req | “db.tb_runstats_req - Statistics Collection Required” on page 563 |
| Database Backup Required                        | db.db_backup_req   | “db.db_backup_req - Database Backup Required” on page 564         |

Table 921. 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 564 |
| HADR Log Delay health indicator          | db.hadr_delay     | “db.hadr_delay - HADR Log Delay” on page 565              |

Table 922. Logging health indicators

| Name                        | Identifier     | Additional Information                                     |
|-----------------------------|----------------|------------------------------------------------------------|
| Log Utilization             | db.log_util    | “db.log_util - Log Utilization” on page 565                |
| Log File System Utilization | db.log_fs_util | “db.log_fs_util - Log File System Utilization” on page 565 |

Table 923. Application concurrency health indicators

| Name                                        | Identifier            | Additional Information                                                            |
|---------------------------------------------|-----------------------|-----------------------------------------------------------------------------------|
| Deadlock Rate                               | db.deadlock_rate      | “db.deadlock_rate - Deadlock Rate” on page 566                                    |
| Lock List Utilization                       | db.locklist_util      | “db.locklist_util - Lock List Utilization” on page 567                            |
| Lock Escalation Rate                        | db.lock_escal_rate    | “db.lock_escal_rate - Lock Escalation Rate” on page 567                           |
| Percentage of Applications Waiting on Locks | db.apps_waiting_locks | “db.apps_waiting_locks - Percentage of Applications Waiting on Locks” on page 568 |

Table 924. 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 569        |
| Package Cache Hit Ratio    | db.pkgcache_hitratio     | "db.pkgcache_hitratio - Package Cache Hit Ratio" on page 569        |
| Shared Workspace Hit Ratio | db.shrworkspace_hitratio | "db.shrworkspace_hitratio - Shared Workspace Hit Ratio" on page 569 |

Table 925. Memory health indicators

| Name                      | Identifier        | Additional Information                                     |
|---------------------------|-------------------|------------------------------------------------------------|
| Monitor Heap Utilization  | db2.mon_heap_util | "db2.mon_heap_util - Monitor Heap Utilization" on page 570 |
| Database Heap Utilization | db.db_heap_util   | "db.db_heap_util - Database Heap Utilization" on page 570  |

Table 926. Federated health indicators

| Name                      | Identifier                 | Additional Information                                             |
|---------------------------|----------------------------|--------------------------------------------------------------------|
| Nickname Status           | db.fed_nicknames_op_status | "db.fed_nicknames_op_status - Nickname Status" on page 571         |
| Data Source Server Status | db.fed_servers_op_status   | "db.fed_servers_op_status - Data Source Server Status" on page 571 |

---

## 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.

## Table space storage health indicators

### Health indicators for DMS table spaces

This table describes which table space health indicators are relevant for a DMS table space based on the characteristics of the table space:

Table 927. 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  | <p>Not a valid configuration. Maximum table space size is only valid for table spaces that have automatic resize enabled.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <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

This health indicator tracks the consumption of storage for the defined database storage paths.

#### 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 calculation of time remaining to fullness in the additional information is a prediction of how much time is remaining until all free space is consumed.

## **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.

**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.

**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” on page 325
- *ts.page\_size* is the value of “tablespace\_page\_size - Table Space Page Size” on page 323
- *ts.max\_size* is the value of “tablespace\_max\_size - Maximum table space size” on page 333

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 calculation of time remaining to fullness in the additional information is a prediction of how much time is remaining until the maximum size has been reached.

## ts.ts\_util - Table Space Utilization

This health indicator tracks the consumption of storage for each DMS table space.

**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” on page 325
- *ts.usable* is the value of “tablespace\_usable\_pages - Usable Pages in Table Space” on page 325

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 calculation of time remaining to fullness in the additional information is a prediction of how much time is remaining until all free space is consumed.

## **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.

### **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 calculation of time remaining to fullness in the additional information is a prediction of how much time is remaining until all free space is consumed.

## **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.

**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.

**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.

**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.

**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 sort heap 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.

**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:

$$\frac{(\text{db.sort\_overflows}_t - \text{db.sort\_overflows}_{t-1})}{(\text{db.total\_sorts}_t - \text{db.total\_sorts}_{t-1})} * 100$$

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.

**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:

$(db.max\_shr\_sort\_mem / sheapthres\_shr) * 100$

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.

**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: 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.

**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: Active, Quiesce pending, Quiesced, or Rollforward. A change from Active to another state may generate an Attention alert.

**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.

**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.

**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.

**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.

**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: Connected, Congested or Disconnected. A change from Connected to another state might generate an Attention alert.

**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.

**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.

**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.

**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.

**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.

**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.

**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:

$$(db.lock\_escal\subscript_t - db.lock\_escal\subscript_{t-1})$$

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.

**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:

$$(db.locks\_waiting / db.apps\_cur\_cons) *100)$$

---

## 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.

**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:

$(1 - (\text{db.cat\_cache\_inserts} / \text{db.cat\_cache\_lookups})) * 100$

### 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.

**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.

**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.

**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:

$$(\text{db2.pool\_cur\_size} / \text{db2.pool\_max\_size}) * 100$$

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.

**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

$$(\text{db.pool\_cur\_size} / \text{db.pool\_max\_size}) * 100$$

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.

**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.

**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.



## Chapter 16. Health monitor interfaces

The following table lists the health monitor interfaces for APIs:

*Table 928. 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 following table lists the health monitor interfaces for CLP commands:

*Table 929. 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 following table lists the health monitor interfaces for SQL functions:

*Table 930. 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 930. Health monitor interfaces: SQL functions (continued)

| Monitoring task                                               | SQL Function       |
|---------------------------------------------------------------|--------------------|
| Table space container level health indicator history snapshot | HEALTH_CONT_HI_HIS |

## Health monitor SQL table functions

The following table lists all of the snapshot table functions. Each table function corresponds to a health snapshot request type.

Table 931. 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                            |
| 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

The following table lists all the supported snapshot request types.

Table 932. Snapshot monitor CLP commands

| Monitor level    | CLP command                 | Information returned                |
|------------------|-----------------------------|-------------------------------------|
| Database manager | get health snapshot for dbm | Database manager level information. |



Table 932. Snapshot monitor CLP commands (continued)

| Monitor level | CLP command                                               | Information returned                                                                                                                                                                                           |
|---------------|-----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 space   | get snapshot for tablespaces on <i>database-alias</i>     | 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 API request types

The following table lists all the supported snapshot request types.

Table 933. 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. |



---

## Part 5. Appendixes



---

## Appendix A. Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Help for DB2 tools
  - 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 hard-copy 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)<sup>®</sup>.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks<sup>®</sup> publications online at [ibm.com](http://www.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/shop/publications/order](http://www.ibm.com/shop/publications/order). English DB2 Version 9.5 manuals in PDF format and translated versions 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 934. DB2 technical information*

| <b>Name</b>                                                                          | <b>Form Number</b> | <b>Available in print</b> |
|--------------------------------------------------------------------------------------|--------------------|---------------------------|
| <i>Administrative API Reference</i>                                                  | SC23-5842-01       | Yes                       |
| <i>Administrative Routines and Views</i>                                             | SC23-5843-01       | No                        |
| <i>Call Level Interface Guide and Reference, Volume 1</i>                            | SC23-5844-01       | Yes                       |
| <i>Call Level Interface Guide and Reference, Volume 2</i>                            | SC23-5845-01       | Yes                       |
| <i>Command Reference</i>                                                             | SC23-5846-01       | Yes                       |
| <i>Data Movement Utilities Guide and Reference</i>                                   | SC23-5847-01       | Yes                       |
| <i>Data Recovery and High Availability Guide and Reference</i>                       | SC23-5848-01       | Yes                       |
| <i>Data Servers, Databases, and Database Objects Guide</i>                           | SC23-5849-01       | Yes                       |
| <i>Database Security Guide</i>                                                       | SC23-5850-01       | Yes                       |
| <i>Developing ADO.NET and OLE DB Applications</i>                                    | SC23-5851-01       | Yes                       |
| <i>Developing Embedded SQL Applications</i>                                          | SC23-5852-01       | Yes                       |
| <i>Developing Java Applications</i>                                                  | SC23-5853-01       | Yes                       |
| <i>Developing Perl and PHP Applications</i>                                          | SC23-5854-01       | No                        |
| <i>Developing User-defined Routines (SQL and External)</i>                           | SC23-5855-01       | Yes                       |
| <i>Getting Started with Database Application Development</i>                         | GC23-5856-01       | Yes                       |
| <i>Getting Started with DB2 installation and administration on Linux and Windows</i> | GC23-5857-01       | Yes                       |
| <i>Internationalization Guide</i>                                                    | SC23-5858-01       | Yes                       |
| <i>Message Reference, Volume 1</i>                                                   | GI11-7855-00       | No                        |
| <i>Message Reference, Volume 2</i>                                                   | GI11-7856-00       | No                        |
| <i>Migration Guide</i>                                                               | GC23-5859-01       | Yes                       |
| <i>Net Search Extender Administration and User's Guide</i>                           | SC23-8509-01       | Yes                       |
| <i>Partitioning and Clustering Guide</i>                                             | SC23-5860-01       | Yes                       |
| <i>Query Patroller Administration and User's Guide</i>                               | SC23-8507-00       | Yes                       |
| <i>Quick Beginnings for IBM Data Server Clients</i>                                  | GC23-5863-01       | No                        |

*Table 934. DB2 technical information (continued)*

| <b>Name</b>                                                                             | <b>Form Number</b> | <b>Available in print</b> |
|-----------------------------------------------------------------------------------------|--------------------|---------------------------|
| <i>Quick Beginnings for DB2 Servers</i>                                                 | GC23-5864-01       | Yes                       |
| <i>Spatial Extender and Geodetic Data Management Feature User's Guide and Reference</i> | SC23-8508-01       | Yes                       |
| <i>SQL Reference, Volume 1</i>                                                          | SC23-5861-01       | Yes                       |
| <i>SQL Reference, Volume 2</i>                                                          | SC23-5862-01       | Yes                       |
| <i>System Monitor Guide and Reference</i>                                               | SC23-5865-01       | Yes                       |
| <i>Troubleshooting Guide</i>                                                            | GI11-7857-01       | No                        |
| <i>Tuning Database Performance</i>                                                      | SC23-5867-01       | Yes                       |
| <i>Visual Explain Tutorial</i>                                                          | SC23-5868-00       | No                        |
| <i>What's New</i>                                                                       | SC23-5869-01       | Yes                       |
| <i>Workload Manager Guide and Reference</i>                                             | SC23-5870-01       | Yes                       |
| <i>pureXML Guide</i>                                                                    | SC23-5871-01       | Yes                       |
| <i>XQuery Reference</i>                                                                 | SC23-5872-01       | No                        |

*Table 935. DB2 Connect-specific technical information*

| <b>Name</b>                                              | <b>Form Number</b> | <b>Available in print</b> |
|----------------------------------------------------------|--------------------|---------------------------|
| <i>Quick Beginnings for DB2 Connect Personal Edition</i> | GC23-5839-01       | Yes                       |
| <i>Quick Beginnings for DB2 Connect Servers</i>          | GC23-5840-01       | Yes                       |
| <i>DB2 Connect User's Guide</i>                          | SC23-5841-01       | Yes                       |

*Table 936. Information Integration technical information*

| <b>Name</b>                                                                                   | <b>Form Number</b> | <b>Available in print</b> |
|-----------------------------------------------------------------------------------------------|--------------------|---------------------------|
| <i>Information Integration: Administration Guide for Federated Systems</i>                    | SC19-1020-01       | Yes                       |
| <i>Information Integration: ASNCLP Program Reference for Replication and Event Publishing</i> | SC19-1018-02       | Yes                       |
| <i>Information Integration: Configuration Guide for Federated Data Sources</i>                | SC19-1034-01       | No                        |
| <i>Information Integration: SQL Replication Guide and Reference</i>                           | SC19-1030-01       | Yes                       |
| <i>Information Integration: Introduction to Replication and Event Publishing</i>              | SC19-1028-01       | Yes                       |

---

## Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation DVD* are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the *DB2 PDF Documentation DVD* can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the *DB2 PDF Documentation DVD* are available in print.

**Note:** The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5>.

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at <http://www.ibm.com/shop/publications/order>. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
  1. Locate the contact information for your local representative from one of the following Web sites:
    - The IBM directory of world wide contacts at [www.ibm.com/planetwide](http://www.ibm.com/planetwide)
    - The IBM Publications Web site at <http://www.ibm.com/shop/publications/order>. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
  2. When you call, specify that you want to order a DB2 publication.
  3. Provide your representative with the titles and form numbers of the books that you want to order. For titles and form numbers, see "DB2 technical library in hardcopy or PDF format" on page 579.

---

## Displaying SQL state help from the command line processor

DB2 returns an `SQLSTATE` value for conditions that could be the result of an SQL statement. `SQLSTATE` help explains the meanings of SQL states and SQL state class codes.

To invoke 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

For DB2 Version 9.5 topics, the DB2 Information Center URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>

For DB2 Version 9 topics, the DB2 Information Center URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>

---

## Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

- To display topics in your preferred language in the Internet Explorer browser:
    1. In Internet Explorer, click the **Tools** → **Internet Options** → **Languages...** button. The Language Preferences window opens.
    2. Ensure your preferred language is specified as the first entry in the list of languages.
      - To add a new language to the list, click the **Add...** button.

**Note:** Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.

    - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.  - 3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.
- To display topics in your preferred language in a Firefox or Mozilla browser:
  1. Select the button in the **Languages** section of the **Tools** → **Options** → **Advanced** dialog. The Languages panel is displayed in the Preferences window.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
    - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
    - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

---

## 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.

Updating your locally-installed DB2 Information Center 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. Non-Administrative and Non-Root DB2 Information Centers always run in stand-alone mode. .
2. Use the Update feature to see what updates are available. If there are updates that you would like to 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, you have to mirror the update site to a local file system 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 Vista, the commands listed below must be run as an administrator. To launch a command prompt or graphical tool with full administrator privileges, right-click on the shortcut and then select **Run as administrator**.

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 on **DB2 Information Center** service and select **Stop**.
  - On Linux, enter the following command:  

```
/etc/init.d/db2icdv95 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 9.5 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/V9.5 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 launches to display the stand-alone Information Center.

3. Click the **Update** button (🔧). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the installation process, check 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 installation directory's doc\bin directory, and run the help\_end.bat file:

```
help_end.bat
```

**Note:** The help\_end batch file contains the commands required to safely terminate the processes that were started with the help\_start batch file. Do not use Ctrl-C or any other method to terminate help\_start.bat.

- On Linux, navigate to the installation directory's doc/bin directory, and run the help\_end script:

```
help_end
```

**Note:** The help\_end script contains the commands required to safely terminate the processes that were started with the help\_start script. Do not use any other method to terminate the help\_start script.

7. Restart the DB2 Information Center.
  - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Start**.
  - On Linux, enter the following command:
 

```
/etc/init.d/db2icdv95 start
```

The updated DB2 Information Center displays the new and updated topics.

---

## DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 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/db2help/>.

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 on 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.

### **“Visual Explain” in *Visual Explain Tutorial***

Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

---

## DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

### **DB2 documentation**

Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

### **DB2 Technical Support Web site**

Refer to the DB2 Technical Support Web site 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 DB2 Technical Support Web site at <http://www.ibm.com/software/data/db2/udb/support.html>

---

## Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal use:** You may reproduce these Publications for your personal, non commercial 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.

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.



---

## Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on 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 give 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 (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, 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 and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This document may provide links or references to non-IBM Web sites and resources. IBM makes no representations, warranties, or other commitments whatsoever about any non-IBM Web sites or third-party resources that may be referenced, accessible from, or linked from this document. A link to a non-IBM Web site does not mean that IBM endorses the content or use of such Web site or

its owner. In addition, IBM is not a party to or responsible for any transactions you may enter into with third parties, even if you learn of such parties (or use a link to such parties) from an IBM site. Accordingly, you acknowledge and agree that IBM is not responsible for the availability of such external sites or resources, and is not responsible or liable for any content, services, products, or other materials on or available from those sites or resources. Any software provided by third parties is subject to the terms and conditions of the license that accompanies that software.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees 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  
Office of the Lab Director  
8200 Warden Avenue  
Markham, Ontario  
L6G 1C7  
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 may contain 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.

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

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries, or both.

|             |         |
|-------------|---------|
| OS/390      | pureXML |
| DB2 Connect | z/OS    |
| Redbooks    | CICS    |
| IBM         | DB2     |
| AIX         | DRDA    |
| OS/400      | RS/6000 |
| ibm.com     |         |

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 are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Microsoft, and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.



---

# Index

## A

- activation time
    - last\_wlm\_reset monitor element 484
  - activities
    - monitor elements
      - act\_total 473
      - activity\_collected 470
      - activity\_id 471
      - activity\_secondary\_id 471
      - activity\_type 472
      - coord\_act\_aborted\_total 476
      - coord\_act\_completed\_total 476
      - coord\_act\_rejected\_total 477
  - Activity Monitor
    - overview 77
    - setup 83
  - administrative views
    - APPL\_PERFORMANCE scenario 80
    - BP\_HITRATIO scenario 82
    - BP\_READ\_IO scenario 82
    - BP\_WRITE\_IO scenario 82
    - LONG\_RUNNING\_SQL scenario 80
    - QUERY\_PREP\_COST scenario 80
    - TOP\_DYNAMIC\_SQL scenario 80
  - agents
    - monitor elements
      - agent\_id 180
      - agent\_id\_holding\_lock 316
      - agent\_pid 204
      - agent\_status 436
      - agent\_sys\_cpu\_time 400
      - agent\_usr\_cpu\_time 400
      - agents\_created\_empty\_pool 213
      - agents\_from\_pool 212
      - agents\_registered 210
      - agents\_registered\_top 211
      - agents\_stolen 214
      - agents\_top 399
      - agents\_waiting\_on\_token 210
      - agents\_waiting\_top 211
      - appl\_priority 195
      - associated agent 205
      - associated\_agents\_top 214
      - coord\_agent\_pid 204
      - coord\_agents\_top 213
      - coordinator agent 205
      - idle\_agents 212
      - locks\_waiting 314
      - max\_agent\_overflows 215
      - num\_agents 399
      - num\_assoc\_agents 215
      - primed agent 205
      - priv\_workspace\_size\_top element 282
      - quiescer\_agent\_id 335
      - rolled\_back\_agent\_id 318
      - subagent 205
  - alert actions
    - health indicators
      - states 544
  - alert thresholds
    - configuration
      - Health Center 543
  - alerts
    - enabling 514
    - resolving
      - GET RECOMMENDATIONS command 535
      - Health Center 536
      - SQL queries 531
    - retrieving recommendations
      - client application 531
  - aliases
    - monitor elements
      - input\_db\_alias element 405
  - API request types
    - health monitor 575
    - snapshot monitor 35
  - application creator monitor element 378
  - applications
    - monitor elements
      - appl\_id 185
      - appl\_id\_holding\_lk 316
      - appl\_id\_oldest\_xact 184
      - appl\_idle\_time 204
      - appl\_name 185
      - appl\_priority 195
      - appl\_priority\_type 196
      - appl\_section\_inserts 285
      - appl\_section\_lookups 285
      - appl\_status 181
      - appls\_cur\_cons 209
      - appls\_in\_db2 210
      - overview 180
      - rolled\_back\_participant\_no 308
      - tpmon\_client\_app 459
  - attributes
    - monitor elements
      - progress\_list\_attr monitor element 417
  - authorization ID
    - monitor element
      - session\_auth\_id element 189
    - monitor elements
      - auth\_id 188
      - execution\_id element 192
      - quiescer\_auth\_id 335
  - authorizations
    - user authorization level monitor element
      - authority\_lvl element 196
  - automatic storage path
    - monitor elements
      - db\_storage\_path 177
      - sto\_path\_free\_sz 178
- ## B
- backup
    - health indicators
      - db.db\_backup\_req 564
    - monitor elements
      - last\_backup 177

- backup (*continued*)
  - requirements
    - health indicator 564
- books
  - printed
    - ordering 582
- buffer pools
  - monitor elements
    - activity 235
    - block\_ios 266
    - bp\_cur\_buffsz 267
    - bp\_id 237
    - bp\_name 263
    - bp\_new\_buffsz 267
    - bp\_pages\_left\_to\_remove 267
    - bp\_tbsp\_use\_count 268
    - buff\_free 232
    - buff\_free\_bottom 233
    - pool\_async\_data\_read\_reqs 258
    - pool\_async\_data\_reads 252
    - pool\_async\_data\_writes 253
    - pool\_async\_index\_read\_reqs 259
    - pool\_async\_index\_reads 255
    - pool\_async\_index\_writes 254
    - pool\_async\_read\_time 257
    - pool\_async\_write\_time 258
    - pool\_async\_xda\_read\_reqs 259
    - pool\_async\_xda\_reads 255
    - pool\_async\_xda\_writes 256
    - pool\_data\_l\_reads 237
    - pool\_data\_p\_reads 239
    - pool\_data\_writes 240
    - pool\_drty\_pg\_steal\_clns 261
    - pool\_drty\_pg\_thrsh\_clns 263
    - pool\_index\_l\_reads 242
    - pool\_index\_p\_reads 243
    - pool\_index\_writes 245
    - pool\_lsn\_gap\_clns 260
    - pool\_no\_victim\_buffer 262
    - pool\_read\_time 250
    - pool\_temp\_data\_l\_reads 238
    - pool\_temp\_data\_p\_reads 240
    - pool\_temp\_index\_l\_reads 243
    - pool\_temp\_index\_p\_reads 244
    - pool\_temp\_xda\_l\_reads 247
    - pool\_temp\_xda\_p\_reads 249
    - pool\_write\_time 251
    - pool\_xda\_l\_reads 246
    - pool\_xda\_p\_reads 248
    - pool\_xda\_writes 250
    - tablespace\_cur\_pool\_id 324
    - tablespace\_next\_pool\_id 324
  - monitoring
    - administrative views 82
- BUFFERPOOLS event type
  - overview 45
- buffers
  - monitor elements
    - num\_log\_data\_found\_in\_buffer 294
- byte order
  - monitor elements
    - byte\_order 407

**C**

- caching
  - stats\_cache\_size monitor element 501
- capturing health snapshots
  - using a client application 524
  - using CLP 523
  - using SQL 522
- catalog cache
  - health indicators
    - db.catcache\_hitratio 569
  - monitor elements
    - cat\_cache\_inserts 273
    - cat\_cache\_lookups 272
    - cat\_cache\_overflows 274
    - cat\_cache\_size\_top 275
    - overview 272
- catalog nodes
  - monitor elements
    - catalog\_node 176
    - catalog\_node\_name 175
- CCSID (coded character set identifier)
  - monitor elements
    - host\_ccsid 436
- channels currently free monitor element
  - FCM (Fast Communications Manager)
    - ch\_free monitor element 234
- client applications
  - capturing health snapshots 524
- client operating platform monitor element
  - client\_platform element 194
- client process ID monitor element
  - client\_pid element 193
- client product and version ID monitor element
  - client\_prdid element 189
- CLP (command line processor)
  - capturing health snapshots 523
  - commands
    - health monitor 574
- code pages
  - monitor elements
    - codepage\_id 183
    - host\_ccsid 436
- commit
  - monitor elements
    - int\_commits element 369
- commit statements attempted monitor element
  - commit\_sql\_stmts element 365
- communication error time monitor element
  - gw\_comm\_error\_time element 458
- communication errors monitor element
  - gw\_comm\_errors element 457
- communication protocol
  - monitor elements
    - client\_protocol 194
- completed progress work units monitor element
  - monitor elements
    - progress\_completed\_units element 417
- con\_response\_time monitor element 457
- connections
  - monitor elements
    - appl\_con\_time 199
    - appls\_cur\_cons 209
    - appls\_in\_db2 210
    - con\_elapsed\_time 457
    - con\_local\_databases 208
    - conn\_complete\_time 200
    - conn\_time 174
    - connection\_status 233
    - connections\_top 200
    - dl\_conns 306

- connections (*continued*)
  - monitor elements (*continued*)
    - gw\_connections\_top 431
    - gw\_cons\_wait\_client 432
    - gw\_cons\_wait\_host 432
    - gw\_cur\_cons 432
    - gw\_total\_cons 431
    - local\_cons 207
    - local\_cons\_in\_exec 208
    - num\_gw\_conn\_switches 216
    - overview 205
    - rem\_cons\_in 206
    - rem\_cons\_in\_exec 206
    - total\_sec\_cons 214
- CONNECTIONS event type
  - overview 45
- containers
  - monitor elements
    - container\_accessible monitor element 339
    - container\_id monitor element 337
    - container\_name monitor element 337
    - container\_total\_pages monitor element 338
    - container\_type monitor element 337
    - container\_usable\_pages monitor element 338
- counters
  - data element type 7
- CPU time
  - monitor elements
    - agent\_sys\_cpu\_time 400
    - agent\_usr\_cpu\_time 400
    - ss\_sys\_cpu\_time 403
    - ss\_usr\_cpu\_time 403
    - stmt\_sys\_cpu\_time 401
    - stmt\_usr\_cpu\_time 401
    - system\_cpu\_time 402
    - tot\_s\_cpu\_time 404
    - tot\_u\_cpu\_time 404
    - user\_cpu\_time 402
- CREATE EVENT MONITOR statement
  - event types 45
- creator monitor element 378
- cursors
  - monitor elements
    - acc\_curs\_blk 362
    - blocking\_cursor 458
    - cursor\_name 377
    - open\_cursors 435
    - open\_loc\_curs 362
    - open\_loc\_curs\_blk 362
    - open\_rem\_curs 360
    - open\_rem\_curs\_blk 361
    - rej\_curs\_blk 361
    - SQL 360
- custom controls
  - accessing 517

**D**

- data
  - element types
    - counters 7
    - overview 5
  - fragmentation
    - overflow\_accesses monitor element 349
- data partitions
  - data partition identifier monitor element 298
- data sources
  - data source name monitor element 460
  - health indicator 571
- data\_partition\_id monitor element 298
- database configuration monitor elements
  - overview 235
- DATABASE event type 45
- database manager monitor elements
  - overview 205
  - server\_db2\_type element 169
- database paths
  - db\_path element monitor element 173
- database system events
  - information collection 47
- database system monitor
  - data organization 5
  - information
    - restricting 15
  - interfaces 507
  - memory requirements 8
  - output 7
  - overview 3
  - sample 507
  - self-describing data stream 7
- database-managed space (DMS)
  - table spaces
    - health indicators 554
- databases
  - aliases
    - application monitor element 190
    - gateway monitor element 430
  - connections
    - connects since database activation monitor element 209
  - monitor elements
    - application 190
    - connects since database activation 209
    - database deactivation timestamp 174
    - database files closed 252
    - database operated on by utility 412
    - gateway 430
- datasource\_name element 460
- db\_heap\_top monitor element
  - description 286
- db.lock\_escal\_rate health indicator 567
- db.locklist\_utilization health indicator 567
- DB2 Connect
  - monitor elements
    - gw\_con\_time 431
    - gw\_cur\_cons 432
    - gw\_exec\_time 433
    - gw\_total\_cons 431
    - overview 430
- DB2 Information Center
  - languages 583
  - updating 584
  - versions 583
  - viewing in different languages 583
- DB2 Performance Counters 101
- db2event.ctl control file 58
- db2perf command
  - resetting database performance values 104
- db2perfi command
  - installing and registering DB2Perf.DLL 101
- db2perfr command
  - registering administrator username and password with DB2 102

- deadlocks
  - db.deadlock\_rate health indicator 566
  - event types 45
  - monitor elements
    - deadlock\_id 307
    - deadlock\_node 307
    - deadlocks 298
    - dl\_conns 306
    - int\_deadlock\_rollbacks 371
    - participant\_no 308
- DELETE statement
  - delete\_sql\_stmts monitor element 462
- descriptors
  - progress\_description monitor element 415
- disconn\_time element 174
- documentation
  - overview 579
  - PDF 579
  - printed 579
  - terms and conditions of use 586
- dynamic buffer pool monitor elements
  - overview 267
- dynamic SQL
  - monitor elements 397

## E

- environment handles
  - comp\_env\_desc monitor element 389
- errors
  - gw\_comm\_errors monitor element 457
- event monitors
  - blocked 59
  - buffers 59
  - creating
    - file 56
    - overview 49
    - partitioned database 61
    - pipe 60
    - table 49
  - data transferring between systems 73
  - database system events 47
  - DEADLOCK WITH DETAILS HISTORY 91
  - elements
    - count 406
    - event\_monitor\_name 408
    - evmon\_activates 410
    - evmon\_flushes 410
    - overview 406
  - event type to logical data group mappings 141
  - file management 58
  - named pipe management 60
  - non-blocked 59
  - output
    - sample 63
    - self-describing data stream 71
  - overview 45
  - records 70
  - table management 52
- event records
  - finding corresponding applications 70
- events
  - monitor elements
    - event\_time 409
    - start\_time 379
    - stop\_time 379

## F

- FCM (Fast Communications Manager)
  - monitor elements
    - buff\_free 232
    - buff\_free\_bottom 233
    - ch\_free 234
    - ch\_free\_bottom 235
    - overview 232
    - total\_buffers\_rcvd 234
    - total\_buffers\_sent 234
- federated databases
  - monitor elements 460
- federated servers
  - monitor elements
    - disconnects element 461
- fetching
  - monitor elements
    - fetch\_count 382
- file event monitors
  - buffering 59
  - creating 56
  - file management 58
  - formatting output from command line 70
- file systems
  - health indicators
    - db.log\_fs\_util 565
  - monitor elements
    - fs\_caching 331
    - fs\_id 179
    - fs\_total\_size 178
    - fs\_type 179
    - fs\_used\_size 178
- files\_closed element 252
- FLUSH EVENT MONITOR statement
  - event types 45
- format
  - health indicator 553

## G

- GET SNAPSHOT command
  - sample output 37, 83
- global health snapshots 528
- global snapshots on partitioned database systems 40
- graphical tools
  - health monitor 529
- gw\_db\_alias element 430

## H

- hash joins
  - monitor elements
    - active\_hash\_joins 228
    - hash\_join\_overflows 229
    - hash\_join\_small\_overflows 230
    - overview 227
    - post\_shrthreshold\_hash\_joins 228
    - post\_threshold\_hash\_joins 228
    - total\_hash\_joins 227
- health alerts
  - enabling 514
  - recommendations
    - retrieving using CLP 531
  - resolving
    - client applications 535
    - SQL queries 531

- Health Center
  - health indicators 511, 547
  - interface 517
  - overview 517, 529
  - status beacon 529
  - tasks 517
- health indicators
  - alert actions
    - combined states 544
  - alerts
    - resolving using SQL 531
    - resolving using the Health Center 536
    - retrieving recommendations 531, 535
  - applications waiting on locks 568
  - catalog cache hit ratio 569
  - collection state-based 511, 547
  - configuration
    - client applications 541
    - Health Center 543
    - overview 537
    - resetting 540
    - retrieving 539
    - updates 539
  - data 522
  - databases
    - heap utilization 570
    - highest severity alert state 562
    - operational state 562
  - db.alert\_state 562
  - db.apps\_waiting\_locks 568
  - db.catcache\_hitratio 569
  - db.database\_heap\_util 570
  - db.db\_auto\_storage\_util 554
  - db.db\_backup\_req 564
  - db.db\_op\_status 562
  - db.deadlock\_rate 566
  - db.fed\_nicknames\_op\_status 571
  - db.fed\_servers\_op\_status 571
  - db.hadr\_delay 565
  - db.hadr\_op\_status 564
  - db.lock\_escal\_rate 567
  - db.locklist\_utilization 567
  - db.log\_fs\_util 565
  - db.log\_util 565
  - db.max\_sort\_shrmem\_util 560
  - db.pkgcache\_hitratio 569
  - db.shrworkspace\_hitratio 569
  - db.sort\_shrmem\_util 559
  - db.spilled\_sorts 560
  - db.tb\_reorg\_req 563
  - db.tb\_runstats\_req 563
  - db2.db2\_alert\_state 561
  - db2.db2\_op\_status 561
  - db2.mon\_heap\_util 570
  - db2.sort\_privmem\_util 558
  - deadlock rate 566
  - DMS table spaces 554
  - format 553
  - instances
    - highest severity alert state 561
    - operational state 561
  - lock escalation rate 567
  - lock list utilization 567
  - logs
    - file system utilization 565
    - space utilization 565
  - monitor heap utilization 570
- health indicators (*continued*)
  - overview 511, 547
  - package cache hit ratio 569
  - process cycle 513
  - shared workspace hit ratio 569
  - sort memory utilization
    - long-term shared 560
    - private 558
    - shared 559
  - sorts that overflowed 560
  - state-based 511, 547
  - summary 551
  - table spaces
    - container operational state 558
    - container utilization 557
    - operational state 558
    - storage utilization 556
  - threshold-based 511, 547
  - ts.ts\_auto\_resize\_status 555
  - ts.ts\_op\_status 558
  - ts.ts\_util 556
  - ts.ts\_util\_auto\_resize 556
  - tsc.tscont\_op\_status 558
  - tsc.utilization 557
- health monitor
  - alerts 537
  - API request types 575
  - CLP commands 574
  - description 511
  - graphical tools 529
  - Health Center 529
  - Health Center Status Beacon 529
  - interfaces 573
  - logical data groups 549
  - recommendation retrieval
    - using client application 535
    - using CLP 531
    - using SQL 531
  - sample output 526
  - SQL table functions 574
  - starting 521
  - stopping 521
  - thresholds 537
- health snapshots
  - capturing
    - using client applications 524
    - using CLP 523
    - using SQL table functions 522
  - global 528
- help
  - configuring language 583
  - SQL statements 582
- High Availability Disaster Recovery (HADR)
  - health indicators
    - db.hadr\_delay 565
    - db.hadr\_op\_status 564
  - monitor elements
    - hadr\_connect\_status 420
    - hadr\_connect\_time 421
    - hadr\_heartbeat 422
    - hadr\_local\_host 422
    - hadr\_local\_service 423
    - hadr\_log\_gap 428
    - hadr\_peer\_window 429
    - hadr\_peer\_window\_end 429
    - hadr\_primary\_log\_file 425
    - hadr\_primary\_log\_lsn 426

## High Availability Disaster Recovery (HADR) *(continued)*

- monitor elements *(continued)*
    - hadr\_primary\_log\_page 426
    - hadr\_remote\_host 423
    - hadr\_remote\_instance 424
    - hadr\_remote\_service 424
    - hadr\_role 418
    - hadr\_standby\_log\_file 427
    - hadr\_standby\_log\_lsn 428
    - hadr\_standby\_log\_page 427
    - hadr\_state 419
    - hadr\_syncmode 420
    - hadr\_timeout 425
  - histograms
    - monitor elements
      - histogram\_type 483
      - number\_in\_bin 485
      - top 497
  - host databases
    - host\_db\_name monitor element 430
    - name monitor element 430
- ## I
- I/O
    - monitor elements
      - num\_log\_part\_page\_io 293
      - num\_log\_write\_io 292
      - num\_pages\_from\_block\_IOs element 266
      - num\_pages\_from\_vectored\_IOs element 265
      - vectored\_ios 265
  - identifiers
    - monitor elements
      - arm\_correlator 473
      - bin\_id 473
      - db\_work\_action\_set\_id 482
      - db\_work\_class\_id 483
      - host\_prdid element 190
      - parent\_activity\_id 485
      - parent\_uow\_id 486
      - sc\_work\_action\_set\_id 489
      - sc\_work\_class\_id 490
      - service\_class\_id 490
      - sql\_req\_id element 411
      - work\_action\_set\_id 498
      - work\_class\_id 499
  - indexes
    - index object pages monitor element 353
    - monitor elements
      - index\_object\_pages element 353
      - reorg\_index\_id monitor element 359
  - Indoubt Transaction Manager
    - overview 11
  - indoubt transactions
    - monitoring 11
  - insert\_timestamp monitor element 380
  - inserting data
    - monitor elements
      - appl\_section\_inserts 285
  - instances
    - operational states
      - health indicator 561
  - int\_rows\_deleted monitor element 349

- ## L
- LOBs (large objects)
    - lob\_object\_pages element 353
  - local databases
    - monitor elements
      - con\_local\_dbases 208
  - location
    - monitor elements
      - db\_location element 176
  - lock list utilization health indicator 567
  - lock modes
    - monitor elements
      - lock\_current\_mode monitor element 312
      - lock\_mode element 301
      - lock\_mode\_requested element 307
  - lock waits
    - monitor elements
      - lock\_wait\_start\_time 315
      - lock\_wait\_time 313
      - lock\_waits 313
  - lock\_escalation element 306
  - locks
    - escalation
      - escalation monitor element 306
      - health indicator 567
    - monitor elements
      - agent\_id\_holding\_lock 316
      - appl\_id\_holding\_lk 316
      - lock\_attributes 309
      - lock\_count 311
      - lock\_escals 299
      - lock\_hold\_count 311
      - lock\_list\_in\_use 297
      - lock\_name monitor element 309
      - lock\_node element 304
      - lock\_object\_name element 304
      - lock\_object\_type element 303
      - lock\_release\_flags monitor element 310
      - lock\_status element 302
      - lock\_timeout\_val element 315
      - lock\_timeouts 305
      - locks\_held 297
      - locks\_held\_top 305
      - locks\_in\_list 309
      - locks\_waiting 314
      - participant\_no\_holding\_lk 308
      - remote\_lock\_time 469
      - remote\_locks 464
      - sequence\_no\_holding\_lk 317
      - stmt\_lock\_timeout 386
      - uow\_lock\_wait\_time 315
      - x\_lock\_escals 300
  - log buffer
    - monitor elements 294
  - log files
    - health indicators
      - db.log\_fs\_util 565
    - monitor elements
      - current\_active\_log 295
      - current\_archive\_log 296
      - first\_active\_log 294
      - hadr\_log\_gap 428
      - hadr\_primary\_log\_file 425
      - hadr\_primary\_log\_page 426
      - hadr\_standby\_log\_file 427
      - hadr\_standby\_log\_page 427
      - last\_active\_log 295



- log files (*continued*)
  - monitor elements (*continued*)
    - log\_read\_time 292
    - log\_reads 288
    - sec\_logs\_allocated element 288
- log sequence number (LSN)
  - monitor elements
    - hadr\_primary\_log\_lsn 426
    - hadr\_standby\_log\_lsn 428
- log spaces
  - health indicators
    - db.log\_util 565
  - monitor elements
    - log\_held\_by\_dirty\_pages 291
    - log\_to\_redo\_for\_recovery 291
    - log\_write\_time 292
    - log\_writes 289
    - sec\_log\_used\_top element 286
    - smallest\_log\_avail\_node 184
    - tot\_log\_used\_top element 287
    - total\_log\_available element 290
    - total\_log\_used element 289
    - uow\_log\_space\_used 289
- logical data groups
  - COLLECT ACTIVITY DATA settings effects 165
  - data organization 5
  - event monitors 144
  - health monitor 549
  - mapping to event types 141
  - snapshot monitor 107
- long data
  - monitor elements
    - long\_object\_pages element 354

## M

- memory
  - health indicators
    - db.sort\_shrmem\_util 559
    - db2.sort\_privmem\_util 558
  - monitor elements
    - comm\_private\_mem 214
    - db\_heap\_top 286
    - lock\_list\_in\_use 297
    - pool\_cur\_size 218
    - pool\_id 216
    - pool\_max\_size 218
    - pool\_secondary\_id 217
    - pool\_watermark 219
  - pool
    - monitor elements 216
- memory requirements
  - database system monitor 8
- Memory Visualizer
  - overview 95
  - using 93
- messages
  - monitor elements
    - message 411
    - message\_time 412
- minimum channels free monitor element
  - FCM (Fast Communications Manager)
    - ch\_free\_bottom monitor element 235
- mon\_heap\_sz database manager configuration parameter 8
- monitor element
  - authorization ID
    - session\_auth\_id element 189

- monitor elements 294
  - acc\_curs\_blk 362
  - accessing
    - overflow\_accesses element 349
  - activation time
    - last\_wlm\_reset 484
  - active\_sorts 225
  - activities
    - act\_total 473
    - activity\_collected 470
    - activity\_id 471
    - activity\_secondary\_id 471
    - activity\_type 472
    - coord\_act\_aborted\_total 476
    - coord\_act\_completed\_total 476
    - coord\_act\_rejected\_total 477
  - agents
    - agent\_id 180
    - agent\_id\_holding\_lock 316
    - agent\_pid 204
    - agent\_status 436
    - agent\_sys\_cpu\_time 400
    - agent\_usr\_cpu\_time 400
    - agents\_created\_empty\_pool 213
    - agents\_from\_pool 212
    - agents\_registered 210
    - agents\_registered\_top 211
    - agents\_stolen 214
    - agents\_top 399
    - agents\_waiting\_on\_token 210
    - agents\_waiting\_top 211
    - appl\_priority 195
    - associated\_agents\_top 214
    - coord\_agent\_pid 204
    - coord\_agents\_top 213
    - idle\_agents 212
    - information 204
    - max\_agent\_overflows 215
    - num\_agents 399
    - num\_assoc\_agents 215
    - overview 205
    - priv\_workspace\_size\_top element 282
    - quiescer\_agent\_id 335
    - rolled\_back\_agent\_id 318
  - aliases
    - client\_db\_alias 190
    - input\_db\_alias element 405
  - applications
    - appl\_id 185
    - appl\_id\_holding\_lk 316
    - appl\_id\_oldest\_xact 184
    - appl\_idle\_time 204
    - appl\_name 185
    - appl\_priority\_type 196
    - appl\_section\_inserts 285
    - appl\_section\_lookups 285
    - appl\_status 181
    - overview 180
    - tpmon\_client\_app 459
  - attributes
    - progress\_list\_attr monitor element 417
  - auth\_id 188
  - authority\_bitmap 197
  - authorization ID
    - execution\_id element 192
  - automatic storage path
    - sto\_path\_free\_sz 178

monitor elements (*continued*)

- binds\_precompiles 372
- blocking\_cursor 458
- blocks\_pending\_cleanup 296
- buffer pools
  - activity 235
  - block\_ios 266
  - bp\_cur\_buffsz 267
  - bp\_id 237
  - bp\_name 263
  - bp\_new\_buffsz 267
  - bp\_pages\_left\_to\_remove 267
  - bp\_tbsp\_use\_count 268
  - buff\_free 232
  - buff\_free\_bottom 233
  - pool\_async\_data\_read\_reqs 258
  - pool\_async\_data\_reads 252
  - pool\_async\_data\_writes 253
  - pool\_async\_index\_read\_reqs 259
  - pool\_async\_index\_reads 255
  - pool\_async\_index\_writes 254
  - pool\_async\_read\_time 257
  - pool\_async\_write\_time 258
  - pool\_async\_xda\_read\_reqs 259
  - pool\_async\_xda\_reads 255
  - pool\_async\_xda\_writes 256
  - pool\_data\_l\_reads 237
  - pool\_data\_p\_reads 239
  - pool\_data\_writes 240
  - pool\_drty\_pg\_steal\_clns 261
  - pool\_drty\_pg\_thrsh\_clns 263
  - pool\_index\_l\_reads 242
  - pool\_index\_p\_reads 243
  - pool\_index\_writes 245
  - pool\_lsn\_gap\_clns 260
  - pool\_no\_victim\_buffer 262
  - pool\_read\_time 250
  - pool\_temp\_data\_l\_reads 238
  - pool\_temp\_data\_p\_reads 240
  - pool\_temp\_index\_l\_reads 243
  - pool\_temp\_index\_p\_reads 244
  - pool\_temp\_xda\_l\_reads 247
  - pool\_temp\_xda\_p\_reads 249
  - pool\_write\_time 251
  - pool\_xda\_l\_reads 246
  - pool\_xda\_p\_reads 248
  - pool\_xda\_writes 250
- buffers
  - num\_log\_data\_found\_in\_buffer 294
- byte order
  - byte\_order 407
- caching
  - stats\_cache\_size 501
- cat\_cache\_inserts 273
- cat\_cache\_lookups 272
- cat\_cache\_overflows 274
- cat\_cache\_size\_top 275
- catalog cache 272
- catalog\_node 176
- catalog\_node\_name 175
- client\_pid element 193
- client\_platform element 194
- client\_prdid element 189
- code pages
  - codepage\_id 183
  - host\_ccsid 436
- comm\_private\_mem 214

monitor elements (*continued*)

- commit
  - int\_commits element 369
- commit\_sql\_stmts element 365
- communication protocol
  - client\_protocol 194
- completed progress work units monitor element
  - progress\_completed\_units element 417
- connections
  - appl\_con\_time 199
  - appls\_cur\_cons 209
  - appls\_in\_db2 210
  - con\_elapsed\_time 457
  - con\_local\_databases 208
  - conn\_complete\_time 200
  - conn\_time 174
  - connection\_status 233
  - connections\_top 200
  - gw\_connections\_top 431
  - gw\_cons\_wait\_client 432
  - gw\_cons\_wait\_host 432
  - gw\_cur\_cons 432
  - gw\_total\_cons 431
  - local\_cons 207
  - local\_cons\_in\_exec 208
  - num\_gw\_conn\_switches 216
  - overview 205
  - rem\_cons\_in 206
  - rem\_cons\_in\_exec 206
  - total\_sec\_cons 214
- container status 336
- containers
  - container\_accessible monitor element 339
  - container\_id monitor element 337
  - container\_name monitor element 337
  - container\_total\_pages monitor element 338
  - container\_type monitor element 337
  - container\_usable\_pages monitor element 338
- coord\_act\_est\_cost\_avg 481
- coord\_act\_exec\_time\_avg 480
- coord\_act\_interarrival\_time\_avg 482
- coord\_act\_lifetime\_avg 479
- coord\_act\_queue\_time\_avg 479
- country\_code
  - replaced with territory\_code 195
- CPU time
  - ss\_sys\_cpu\_time 403
  - ss\_usr\_cpu\_time 403
  - stmt\_sys\_cpu\_time 401
  - stmt\_usr\_cpu\_time 401
  - system\_cpu\_time 402
  - tot\_s\_cpu\_time 404
  - tot\_u\_cpu\_time 404
  - user\_cpu\_time 402
- CPU usage 400
- cursor\_name 377
- data organization 5
- data\_partition\_id 298
- database and application activity 296
- database configuration 235
- database connections
  - total\_cons element 209
- database heap 286
- database identification and status 172
- database manager
  - server\_db2\_type element 169
- database manager configuration 205

monitor elements (*continued*)

- database paths
  - db\_path element 173
- database system 167
- db\_heap\_top 286
- db\_storage\_path 177
- DB2 Connect
  - gw\_con\_time 431
  - gw\_exec\_time 433
  - overview 430
- deadlocks
  - deadlock\_id 307
  - deadlock\_node 307
  - deadlocks 298
  - dl\_conns 306
  - int\_deadlock\_rollbacks 371
  - overview 297
- DELETE statement
  - delete\_sql\_stmts element 462
- deleting
  - int\_rows\_deleted element 349
- descriptors
  - progress\_description element 415
- Dynamic buffer pool
  - overview 267
- dynamic SQL 397
- environment handles
  - comp\_env\_desc element 389
- errors
  - gw\_comm\_errors element 457
- event monitors
  - count 406
  - event\_monitor\_name 408
  - evmon\_activates 410
  - evmon\_flushes 410
  - list 144
  - overview 406
- events
  - event\_time element 409
  - start\_time element 379
  - stop\_time element 379
- executing
  - act\_exec\_time 472
- fabrications
  - stats\_fabricate\_time 504
  - stats\_fabrications 502
- fast communications manager 232
- FCM (Fast Communications Manager)
  - ch\_free monitor element 234
  - ch\_free\_bottom monitor element 235
  - total\_buffers\_rcvd element 234
  - total\_buffers\_sent element 234
- federated database systems 460
- federated servers
  - disconnects element 461
- fetching
  - fetch\_count 382
- file systems
  - fs\_caching 331
  - fs\_id 179
  - fs\_total\_size 178
  - fs\_type 179
  - fs\_used\_size 178
- gw\_comm\_error\_time element 458
- HADR
  - HADR peer window 429
  - HADR peer window end 429

monitor elements (*continued*)

- HADR (high availability disaster recovery)
  - hadr\_connect\_status 420
  - hadr\_connect\_time 421
  - hadr\_heartbeat 422
  - hadr\_local\_host 422
  - hadr\_local\_service 423
  - hadr\_log\_gap 428
  - hadr\_primary\_log\_file 425
  - hadr\_primary\_log\_lsn 426
  - hadr\_primary\_log\_page 426
  - hadr\_remote\_host 423
  - hadr\_remote\_instance 424
  - hadr\_remote\_service 424
  - hadr\_role 418
  - hadr\_standby\_log\_file 427
  - hadr\_standby\_log\_lsn 428
  - hadr\_standby\_log\_page 427
  - hadr\_state 419
  - hadr\_syncmode 420
  - hadr\_timeout 425
  - overview 418
- hash joins
  - active\_hash\_joins 228
  - hash\_join\_overflows 229
  - hash\_join\_small\_overflows 230
  - overview 227
  - post\_shrthreshold\_hash\_joins 228
  - post\_threshold\_hash\_joins 228
  - total\_hash\_joins 227
- histograms
  - histogram\_type 483
  - number\_in\_bin 485
  - top 497
- host databases
  - host\_db\_name element 430
- I/O
  - num\_log\_part\_page\_io 293
  - num\_log\_write\_io 292
  - num\_pages\_from\_block\_IOs element 266
  - num\_pages\_from\_vectorized\_IOs element 265
  - vectorized\_ios 265
- identifiers
  - arm\_correlator 473
  - bin\_id 473
  - db\_work\_action\_set\_id 482
  - db\_work\_class\_id 483
  - host\_prdid element 190
  - parent\_activity\_id 485
  - parent\_uow\_id 486
  - sc\_work\_action\_set\_id 489
  - sc\_work\_class\_id 490
  - service\_class\_id 490
  - sql\_req\_id element 411
  - work\_action\_set\_id 498
  - work\_class\_id 499
- inbound\_bytes\_received element 437
- inbound\_bytes\_sent element 439
- inbound\_comm\_address element 437
- indexes
  - index\_object\_pages element 353
- insert\_timestamp 380
- intra-query parallelism 399
- is\_system\_appl 191
- LOBs (large objects)
  - lob\_object\_pages element 353

monitor elements (*continued*)

- location
  - db\_location element 176
- lock modes
  - lock\_current\_mode monitor element 312
  - lock\_mode element 301
  - lock\_mode\_requested element 307
- lock waits
  - lock\_wait\_time 313
  - lock\_waits 313
  - overview 312
- locks
  - lock\_attributes 309
  - lock\_count 311
  - lock\_escals 299
  - lock\_hold\_count 311
  - lock\_list\_in\_use 297
  - lock\_name monitor element 309
  - lock\_node element 304
  - lock\_object\_name element 304
  - lock\_object\_type element 303
  - lock\_release\_flags monitor element 310
  - lock\_status element 302
  - lock\_timeout\_val element 315
  - lock\_timeouts 305
  - locks\_held 297
  - locks\_held\_top 305
  - locks\_in\_list 309
  - locks\_waiting 314
  - overview 297
  - participant\_no\_holding\_lk 308
  - remote\_lock\_time 469
  - remote\_locks 464
  - sequence\_no\_holding\_lk 317
  - stmt\_lock\_timeout 386
  - uow\_lock\_wait\_time 315
  - x\_lock\_escals 300
- log buffers
  - num\_log\_buffer\_full 294
- log files
  - current\_active\_log 295
  - current\_archive\_log 296
  - first\_active\_log 294
  - last\_active\_log 295
  - log\_read\_time 292
  - log\_reads 288
  - sec\_logs\_allocated element 288
- log spaces
  - log\_held\_by\_dirty\_pages 291
  - log\_to\_redo\_for\_recovery 291
  - log\_write\_time 292
  - log\_writes 289
  - sec\_log\_used\_top element 286
  - smallest\_log\_avail\_node 184
  - tot\_log\_used\_top element 287
  - total\_log\_available element 290
  - total\_log\_used element 289
  - uow\_log\_space\_used 289
- logging 286
- long data
  - long\_object\_pages element 354
- memory pool 216
- messages
  - message 411
- names
  - db\_name element 172
  - dcs\_db\_name element 430

monitor elements (*continued*)

- names (*continued*)
  - service\_subclass\_name 491
  - service\_superclass\_name 491
  - work\_action\_set\_name 499
  - work\_class\_name 499
- network time
  - max\_network\_time\_1\_ms 450
  - max\_network\_time\_100\_ms 452
  - max\_network\_time\_16\_ms 451
  - max\_network\_time\_4\_ms 451
  - max\_network\_time\_500\_ms 452
  - max\_network\_time\_gt500\_ms 453
- network\_time\_bottom 453
- network\_time\_top 453
- nicknames
  - create\_nickname element 463
  - create\_nickname\_time element 468
- nodes
  - coord\_node monitor element 199
  - node\_number element 198
  - num\_nodes\_in\_db2\_instance 406
  - ss\_node\_number element 392
- non-buffered I/O activity 268
- num\_db\_storage\_paths 177
- num\_indoubt\_trans 312
- num\_log\_part\_page\_io 293
- num\_log\_read\_io 293
- num\_log\_write\_io 292
- num\_nodes\_in\_db2\_instance 406
- num\_transmissions 456
- num\_transmissions\_group 456
- numbers
  - progress\_list\_cur\_seq\_num element 415
  - ss\_number element 392
- OLAP functions 230, 231, 232
- On-Line Analytical Processing (OLAP)
  - overview 230
- open\_cursors 435
- open\_loc\_curs 362
- open\_loc\_curs\_blk 362
- open\_rem\_curs 360
- open\_rem\_curs\_blk 361
- operations
  - direct\_read\_reqs element 270
  - direct\_read\_time element 271
  - direct\_reads element 268
  - direct\_write\_reqs element 270
  - direct\_write\_time element 271
  - direct\_writes element 269
  - stmt\_operation element 374
- outbound bytes
  - max\_data\_sent\_1024 443
  - max\_data\_sent\_128 441
  - max\_data\_sent\_16384 447
  - max\_data\_sent\_2048 444
  - max\_data\_sent\_256 442
  - max\_data\_sent\_31999 448
  - max\_data\_sent\_4096 445
  - max\_data\_sent\_512 443
  - max\_data\_sent\_64000 449
  - max\_data\_sent\_8192 446
  - max\_data\_sent\_gt64000 449
- outbound bytes received
  - max\_data\_received\_1024 444
  - max\_data\_received\_128 441
  - max\_data\_received\_16384 447

- monitor elements (*continued*)
  - outbound bytes received (*continued*)
    - max\_data\_received\_2048 445
    - max\_data\_received\_256 442
    - max\_data\_received\_31999 448
    - max\_data\_received\_4096 446
    - max\_data\_received\_512 443
    - max\_data\_received\_64000 449
    - max\_data\_received\_8192 446
    - max\_data\_received\_gt64000 450
    - outbound\_bytes\_received 438
    - outbound\_bytes\_received\_bottom 440
    - outbound\_bytes\_received\_top 440
  - outbound bytes sent
    - outbound\_bytes\_sent 438
    - outbound\_bytes\_sent\_bottom 440
    - outbound\_bytes\_sent\_top 439
  - outbound communication
    - outbound\_appl\_id 191
    - outbound\_comm\_address 437
    - outbound\_comm\_protocol 436
  - outbound sequence
    - outbound\_sequence\_no 192
  - overflow records
    - first\_overflow\_time element 407
    - last\_over\_flow time element 407
  - overflow\_accesses 349
  - package cache 275
    - pkg\_cache\_inserts 277
    - pkg\_cache\_lookups 276
    - pkg\_cache\_num\_overflow 278
    - pkg\_cache\_size\_top 278
  - package names
    - package\_name element 375
  - packages
    - package\_version\_id element 376
  - pages
    - data\_object\_pages element 352
  - parallelism
    - degree\_parallelism 399
  - participant\_no 308
  - partition information
    - partition\_number monitor element 412
  - partitions
    - coord\_partition\_num 478
  - pass-through
    - passthru 463
    - passthru\_time 468
  - pool\_cur\_size 218
  - pool\_id 216
  - pool\_max\_size 218
  - pool\_secondary\_id 217
  - pool\_watermark 219
  - prefetching
    - unread\_prefetch\_pages element 264
  - priv\_workspace\_num\_overflows element 283
  - progress\_work\_metric element 416
  - queries
    - query\_card\_estimate 383
    - query\_cost\_estimate 384
    - queue\_assignments\_total 486
    - queue\_size\_top 487
    - queue\_time\_total 487
    - select\_time 465
  - quiescer
    - quiescer\_auth\_id 335
    - quiescer\_obj\_id 336

- monitor elements (*continued*)
  - quiescer (*continued*)
    - quiescer\_state 336
    - quiescer\_ts\_id 336
  - ranges
    - bottom 474
    - range\_adjustment element 341
    - range\_container\_id element 342
    - range\_end\_stripe element 341
    - range\_max\_extent element 340
    - range\_max\_page\_number element 340
    - range\_num\_containers 341
    - range\_number element 340
    - range\_offset element 342
    - range\_start\_stripe element 341
    - range\_stripe\_set\_number element 339
  - Real-time statistics
    - overview 501
  - rebinding
    - int\_auto\_rebinds element 369
  - records
    - partial\_record element 408
  - rej\_curs\_blk 361
  - reoptimization
    - stmt\_value\_isreopt 391
  - reorg\_completion element 358
  - reorg\_long\_tbsp\_id 359
  - reorg\_tbsp\_id 359
  - reorganization
    - page\_reorgs element 352
    - reorg\_current\_counter element 357
    - reorg\_max\_phase element 357
    - reorg\_phase monitor element 356
    - reorg\_phase\_start element 356
    - reorg\_rows\_compressed monitor element 359
    - reorg\_rows\_rejected\_for\_compression monitor element 360
    - reorg\_start element 358
    - reorg\_status element 355
    - reorg\_type element 355
  - request\_exec\_time\_avg 481
  - response time
    - delete\_time element 467
    - host\_response\_time element 455
    - insert\_time element 466
  - roll-forward recovery
    - rf\_log\_num 319
    - rf\_status 320
    - rf\_timestamp 319
    - rf\_type 319
  - rollback
    - rollback\_sql\_stmts 366
    - rolled\_back\_appl\_id 317
    - rolled\_back\_participant\_no 308
    - rolled\_back\_sequence\_no 318
  - rollbacks
    - int\_rollbacks 370
  - rollforward recovery
    - overview 318
  - rows
    - int\_rows\_inserted element 351
    - int\_rows\_updated element 350
    - rows\_deleted element 345
    - rows\_fetched 487
    - rows\_inserted element 345
    - rows\_modified 488
    - rows\_read element 348

- monitor elements (*continued*)
  - rows (*continued*)
    - rows\_returned 488
    - rows\_selected element 346
    - rows\_updated element 346
    - rows\_written element 347
    - sp\_rows\_selected element 465
  - RUNSTATS utility
    - async\_runstats 503
    - sync\_runstats 502
    - sync\_runstats\_time 504
  - section\_env 490
  - sections
    - priv\_workspace\_section\_inserts element 284
    - priv\_workspace\_section\_lookups element 283
    - section\_number element 377
  - sequences
    - progress\_seq\_num element 415
    - sequence\_no 187
  - server identification and status 168
  - servers
    - product\_name 171
    - server\_instance\_name 168
    - server\_platform 171
    - server\_prdid 169
    - server\_version 170
  - service levels
    - service\_level 170
  - shared workspace
    - shr\_workspace\_num\_overflows 280
    - shr\_workspace\_section\_inserts 281
    - shr\_workspace\_section\_lookups 280
    - shr\_workspace\_size\_top 279
  - snapshot monitor logical data groups 111
  - snapshot monitoring 404
  - snapshots
    - time\_stamp element 406
  - sort 219
  - sorting
    - piped\_sorts\_accepted element 222
    - piped\_sorts\_requested element 221
    - post\_shrthreshold\_sorts\_monitor element 221
    - post\_threshold\_sorts element 220
    - sort\_heap\_allocated element 220
    - sort\_heap\_top\_monitor element 225
    - sort\_overflows element 224
    - sort\_shrheap\_allocated\_monitor element 226
    - sort\_shrheap\_top\_monitor element 226
    - total\_sorts element 222
  - SQL communications area (SQLCA)
    - sqlca 383
  - SQL cursors 360
  - SQL operations
    - elapsed\_exec\_time element 454
  - SQL statement activity 363
  - SQL statement details 373
  - SQL statements
    - ddl\_sql\_stmts element 368
    - dynamic\_sql\_stmts element 364
    - failed\_sql\_stmts element 364
    - insert\_sql\_stmts element 461
    - num\_compilation element 397
    - num\_executions element 397
    - select\_sql\_stmts element 367
    - sql\_chains 434
    - sql\_reqs\_since\_commit element 372
    - sql\_stmts 433
- monitor elements (*continued*)
  - SQL statements (*continued*)
    - static\_sql\_stmts element 363
    - stmt\_pkgcache\_id element 389
    - stmt\_query\_id element 388
    - stmt\_sorts element 381
    - stmt\_source\_id element 388
    - stmt\_text element 381
    - stmt\_value\_data element 390
    - stmt\_value\_index element 391
    - stmt\_value\_isnull element 390
    - stmt\_value\_type element 390
    - total\_exec\_time element 398
    - uid\_sql\_stmts element 367
  - SQL workspaces 279
  - statements
    - prep\_time\_best element 398
    - prep\_time\_worst element 398
    - stmt\_first\_use\_time element 385
    - stmt\_history\_id element 385
    - stmt\_history\_list\_size element 391
    - stmt\_invocation\_id element 387
    - stmt\_isolation element 386
    - stmt\_last\_use\_time 386
    - stmt\_nest\_level element 387
    - stmt\_node\_number element 372
    - stmt\_type element 374
  - status
    - db\_status element 175
    - db2\_status element 171
    - dcs\_appl\_status element 435
    - ss\_status element 393
  - storage paths
    - num\_db\_storage\_paths 177
  - stored procedures
    - stored\_proc\_time element 469
    - stored\_procs element 464
  - stripe sets
    - container\_stripe\_set\_monitor element 338
  - subsection details 392
  - table activity 342
  - table reorganization 355
    - reorg\_end element 358
  - table spaces
    - activity 320
    - quiescer activity 335
    - range status 339
    - tablespace\_auto\_resize\_enabled 332
    - tablespace\_content\_type 322
    - tablespace\_cur\_pool\_id 324
    - tablespace\_current\_size 332
    - tablespace\_extent\_size 323
    - tablespace\_free\_pages 326
    - tablespace\_id 320
    - tablespace\_increase\_size 333
    - tablespace\_increase\_size\_percent 334
    - tablespace\_initial\_size 332
    - tablespace\_last\_resize\_failed 335
    - tablespace\_last\_resize\_time 334
    - tablespace\_max\_size 333
    - tablespace\_min\_recovery\_time 330
    - tablespace\_name 320
    - tablespace\_next\_pool\_id 324
    - tablespace\_num\_containers 331
    - tablespace\_num\_quiescers 329
    - tablespace\_num\_ranges 331
    - tablespace\_page\_size 323

monitor elements (*continued*)

- table spaces (*continued*)
  - tablespace\_page\_top 326
  - tablespace\_pending\_free\_pages 326
  - tablespace\_prefetch\_size 323
  - tablespace\_rebalancer\_extents\_processed 328
  - tablespace\_rebalancer\_extents\_remaining 328
  - tablespace\_rebalancer\_last\_extent\_moved 328
  - tablespace\_rebalancer\_mode 327
  - tablespace\_rebalancer\_priority 329
  - tablespace\_rebalancer\_restart\_time 327
  - tablespace\_rebalancer\_start\_time 327
  - tablespace\_state 322
  - tablespace\_state\_change\_object\_id 330
  - tablespace\_state\_change\_ts\_id 330
  - tablespace\_total\_pages 325
  - tablespace\_type 321
  - tablespace\_usable\_pages 325
  - tablespace\_used\_pages 325
  - tablespace\_using\_auto\_storage 331
  - ts\_name 319
- tables
  - table\_file\_id element 351
  - table\_name element 343
  - table\_schema element 344
  - table\_type element 342
- territory\_code 195
- thresholds
  - num\_threshold\_violations 484
  - threshold\_action 493
  - threshold\_domain 493
  - threshold\_maxvalue 494
  - threshold\_name 494
  - threshold\_predicate 494
  - threshold\_queue\_size 495
  - thresholdid 495
- time
  - prefetch\_wait\_time element 264
  - prep\_time 486
  - progress\_start\_time element 416
  - ss\_exec\_time element 393
  - stmt\_elapsed\_time element 380
  - time\_completed 496
  - time\_created 496
  - time\_of\_violation 496
  - time\_started 497
  - total\_sort\_time element 223
- time zones
  - time\_zone\_disp element 172
- timestamps
  - activate\_timestamp 470
  - db\_conn\_time 174
  - db2start\_time 168
  - last\_backup 177
  - last\_reset 405
  - lock\_wait\_start\_time 315
  - message\_time 412
  - statistics\_timestamp 492
  - status\_change\_time 183
  - stmt\_start 378
  - stmt\_stop 379
- tokens
  - consistency\_token monitor element 376
  - corr\_token monitor element 193
- total\_hash\_loops element 229
- tq\_cur\_send\_spills 395
- tq\_id\_waiting\_on 397

monitor elements (*continued*)

- tq\_max\_send\_spills 396
- tq\_node\_waited\_for 394
- tq\_rows\_read 395
- tq\_rows\_written 396
- tq\_tot\_send\_spills 394
- tq\_wait\_for\_any 394
- transaction processing
  - tpmon\_acc\_str 460
  - tpmon\_client\_userid 459
  - tpmon\_client\_wkstn 459
- transaction processor monitoring 458
- transactions
  - num\_indoubt\_trans 312
  - xid monitor element 454
- units of work (UOW)
  - prev\_uow\_stop\_time 201
  - progress\_total\_units element 416
  - uow\_comp\_status 203
  - uow\_elapsed\_time 202
  - uow\_id 497
  - uow\_start\_time 201
  - uow\_status 203
  - uow\_stop\_time 202
- updates
  - update\_sql\_stmts element 462
- utilities 412
  - utility\_description element 414
  - utility\_id element 412
  - utility\_invoker\_type element 414
  - utility\_priority element 413
  - utility\_type 413
- vectored\_ios 265
- watermarks
  - concurrent\_act\_top 474
  - concurrent\_connection\_top 475
  - concurrent\_wlo\_act\_top 475
  - concurrent\_wlo\_top 475
  - coord\_act\_lifetime\_top 477
  - cost\_estimate\_top 478
  - rows\_returned\_top 489
  - temp\_tablespace\_top 492
- Workload management
  - overview 470
- workloads
  - wlo\_completed\_total 498
  - workload\_id 500
  - workload\_name 500
  - workload\_occurrence\_id 501
- xquery\_stmts 373
- monitor heap
  - health indicators
    - db2.mon\_heap\_util 570
- monitor switches
  - description 15
  - setting from a client application 18
  - setting from the CLP 17
- monitoring
  - buffer pool efficiency
    - administrative views 82
  - capturing a snapshot from client applications 34
  - capturing a snapshot from the command line 31
  - capturing a snapshot using SQL 22, 30
    - with file access 26
    - with SNAP\_WRITE\_FILE 24
  - data partitions 84
  - database activity 77, 83



- monitoring (*continued*)
  - database events 45
    - event types 45
    - overview 3
    - sample output 63
  - database recovery 318
  - databases
    - overview 3
  - health monitor 511, 521
  - memory components 95
  - open access to monitor data
    - capturing snapshot information to a file 24
    - retrieving snapshot information from a file 26
    - SYSMON authority 21
  - runtime rollback process 83
  - snapshot
    - API request types 35
    - CLP commands 31
    - overview 3
    - system monitor 3
- most recent response time for connect monitor element 457

## N

- names
  - monitor elements
    - db\_name element 172
    - dcs\_db\_name element 430
    - service\_subclass\_name 491
    - service\_superclass\_name 491
    - work\_action\_set\_name 499
    - work\_class\_name 499
- network time
  - monitor elements
    - max\_network\_time\_1\_ms 450
    - max\_network\_time\_100\_ms 452
    - max\_network\_time\_16\_ms 451
    - max\_network\_time\_4\_ms 451
    - max\_network\_time\_500\_ms 452
    - max\_network\_time\_gt500\_ms 453
    - network\_time\_bottom 453
    - network\_time\_top 453
- nicknames
  - health indicator 571
  - monitor elements
    - create\_nickname element 463
    - create\_nickname\_time element 468
- nodes
  - monitor elements
    - coord\_node monitor element 199
    - node\_number element 198
    - num\_nodes\_in\_db2\_instance 406
    - ss\_node\_number element 392
- notices 589
- num\_indoubt\_trans element 312
- num\_log\_data\_found\_in\_buffer element 294
- num\_log\_part\_page\_io element 293
- num\_log\_read\_io element 293
- num\_log\_write\_io element 292
- num\_transmissions element 456
- num\_transmissions\_group element 456
- num\_vectored\_IOs element 265
- numbers
  - monitor elements
    - progress\_list\_cur\_seq\_num element 415
    - ss\_number element 392

## O

- objects
  - performance on Windows 102
- OLAP functions
  - monitor elements 230, 231, 232
- Online Analytical Processing (OLAP)
  - monitor elements
    - overview 230
- operation monitor element 374
- operations
  - monitor elements
    - direct\_read\_reqs element 270
    - direct\_read\_time element 271
    - direct\_reads element 268
    - direct\_write\_reqs element 270
    - direct\_write\_time element 271
    - direct\_writes element 269
    - stmt\_operation element 374
- optimization
  - monitor elements
    - stmt\_value\_isreopt 391
- ordering DB2 books 582
- outbound bytes received
  - monitor elements
    - max\_data\_received\_1024 444
    - max\_data\_received\_128 441
    - max\_data\_received\_16384 447
    - max\_data\_received\_2048 445
    - max\_data\_received\_256 442
    - max\_data\_received\_31999 448
    - max\_data\_received\_4096 446
    - max\_data\_received\_512 443
    - max\_data\_received\_64000 449
    - max\_data\_received\_8192 446
    - max\_data\_received\_gt64000 450
    - outbound\_bytes\_received 438
    - outbound\_bytes\_received\_bottom 440
    - outbound\_bytes\_received\_top 440
- outbound bytes sent
  - monitor elements
    - max\_data\_sent\_1024 443
    - max\_data\_sent\_128 441
    - max\_data\_sent\_16384 447
    - max\_data\_sent\_2048 444
    - max\_data\_sent\_256 442
    - max\_data\_sent\_31999 448
    - max\_data\_sent\_4096 445
    - max\_data\_sent\_512 443
    - max\_data\_sent\_64000 449
    - max\_data\_sent\_8192 446
    - max\_data\_sent\_gt64000 449
    - outbound\_bytes\_sent 438
    - outbound\_bytes\_sent\_bottom 440
    - outbound\_bytes\_sent\_top 439
- outbound communication
  - monitor elements
    - outbound\_appl\_id 191
    - outbound\_comm\_address 437
    - outbound\_comm\_protocol 436
    - outbound\_sequence\_no 192
- overflow records
  - monitor elements
    - first\_overflow\_time element 407
    - last\_over\_flow time element 407



## P

- package cache
  - db.pkgcache\_hitratio 569
  - monitor elements
    - pkg\_cache\_inserts 277
    - pkg\_cache\_lookups 276
    - pkg\_cache\_num\_overflow 278
    - pkg\_cache\_size\_top 278
- packages
  - names
    - package\_name monitor element 375
    - package\_version\_id monitor element 376
- pages
  - bp\_pages\_left\_to\_remove monitor element 267
  - data\_object\_pages monitor element 352
  - removing
    - bp\_pages\_left\_to\_remove monitor element 267
- parallelism
  - monitor elements
    - degree\_parallelism 399
    - intra-query 399
- partial\_record monitor element 408
- partition\_number monitor element 412
- partitioned database environments
  - coord\_partition\_num monitor element 478
  - event monitoring 61
  - global snapshots 40
- partitioned tables
  - reorganizing 84
- pass-through
  - monitor elements
    - passthru\_time 468
    - passthru 463
- performance
  - information
    - displaying 102
    - enabling remote access 102
  - remote databases 103
  - resetting values 104
  - Windows
    - monitoring tools 101
    - Performance Monitor objects 102
- pipe event monitors
  - creating 60
  - formatting output from command line 70
  - named pipe management 60
- piped\_sorts\_accepted monitor element 222
- piped\_sorts\_requested monitor element 221
- post\_shrthreshold\_sorts monitor element 221
- post\_threshold\_sorts monitor element 220
- prefetching
  - monitor elements
    - unread\_prefetch\_pages 264
- priv\_workspace\_num\_overflows monitor element
  - description 283
- priv\_workspace\_section\_inserts monitor element
  - description 284
- priv\_workspace\_section\_lookups monitor element
  - description 283
- priv\_workspace\_size\_top monitor element
  - description 282
- problem determination
  - information available 586
  - tutorials 586
- processes
  - monitor elements
    - agent\_pid 204

- progress\_description monitor element 415
- progress\_seq\_num monitor element 415
- progress\_start\_time monitor element 416
- progress\_work\_metric monitor element 416

## Q

- queries
  - monitor elements
    - query\_card\_estimate 383
    - query\_cost\_estimate 384
    - queue\_assignments\_total 486
    - queue\_size\_top 487
    - queue\_time\_total 487
    - select\_time 465
- quiescer
  - monitor elements
    - quiescer\_auth\_id 335
    - quiescer\_obj\_id 336
    - quiescer\_state 336
    - quiescer\_ts\_id 336

## R

- range adjustment monitor element 341
- range container monitor element 342
- range number monitor element 340
- range offset monitor element 342
- range\_num\_containers monitor element 341
- ranges
  - monitor elements
    - bottom 474
    - range\_adjustment element 341
    - range\_container\_id element 342
    - range\_end\_stripe element 341
    - range\_max\_extent element 340
    - range\_max\_page\_number element 340
    - range\_num\_containers 341
    - range\_number element 340
    - range\_offset element 342
    - range\_start\_stripe element 341
    - range\_stripe\_set\_number element 339
- real-time statistics
  - monitor elements
    - overview 501
    - stats\_fabricate\_time 504
    - stats\_fabrications 502
- rebalancing
  - monitor elements
    - tablespace\_rebalancer\_extents\_processed 328
    - tablespace\_rebalancer\_extents\_remaining 328
    - tablespace\_rebalancer\_last\_extent\_moved 328
    - tablespace\_rebalancer\_mode 327
    - tablespace\_rebalancer\_priority 329
    - tablespace\_rebalancer\_restart\_time 327
    - tablespace\_rebalancer\_start\_time 327
- rebinding
  - monitor elements
    - int\_auto\_rebinds element 369
- records
  - monitor elements
    - partial\_record element 408
- recovery
  - monitor elements
    - log\_to\_redo\_for\_recovery 291

- remote
  - performance 103
- reoptimization
  - monitor elements
    - stmt\_value\_isreopt 391
- reorg\_index\_id monitor element 359
- reorganization
  - health indicators
    - db.tb\_reorg\_req 563
  - monitor elements
    - page\_reorgs element 352
    - reorg\_current\_counter element 357
    - reorg\_max\_counter element 357
    - reorg\_max\_phase element 357
    - reorg\_phase monitor element 356
    - reorg\_phase\_start element 356
    - reorg\_rows\_compressed monitor element 359
    - reorg\_rows\_rejected\_for\_compression monitor element 360
    - reorg\_start element 358
    - reorg\_status element 355
    - reorg\_type element 355
  - reorganize phase monitor element 356
  - request identifier for sql statement monitor element 411
  - response time
    - monitor elements
      - delete\_time element 467
      - host\_response\_time element 455
      - insert\_time element 466
  - roll-forward recovery
    - monitor elements
      - overview 318
      - rf\_log\_num 319
      - rf\_status 320
      - rf\_timestamp 319
      - rf\_type 319
      - tablespace\_min\_recovery\_time 330
      - ts\_name 319
  - rollbacks
    - monitor elements
      - int\_deadlock\_rollbacks 371
      - int\_rollbacks 370
      - rf\_status 320
      - rollback\_sql\_stmts 366
      - rolled\_back\_agent\_id 318
      - rolled\_back\_appl\_id 317
      - rolled\_back\_participant\_no 308
      - rolled\_back\_sequence\_no 318
    - monitoring progress 83
  - rows
    - monitor elements
      - int\_rows\_inserted 351
      - int\_rows\_updated 350
      - rows\_deleted 345
      - rows\_fetched 487
      - rows\_inserted 345
      - rows\_modified 488
      - rows\_read 348
      - rows\_returned 488
      - rows\_returned\_top 489
      - rows\_selected 346
      - rows\_updated 346
      - rows\_written 347
      - sp\_rows\_selected 465
    - rows compressed monitor element 359
    - rows deleted monitor element 345
    - rows inserted monitor element 345

- rows read monitor element 348
- rows rejected for compression monitor element 360
- rows returned by stored procedures monitor element 465
- rows selected monitor element 346
- rows updated monitor element 346
- rows written monitor element 347
- RUNSTATS utility
  - monitor elements
    - async\_runstats 503
    - sync\_runstats 502
    - sync\_runstats\_time 504

## S

- schemas
  - tables
    - table\_schema element 344
- secondary logs allocated currently monitor element 288
- section number monitor element 377
- sections
  - monitor elements
    - appl\_section\_inserts 285
    - appl\_section\_lookups 285
    - priv\_workspace\_section\_inserts element 284
    - priv\_workspace\_section\_lookups element 283
    - section\_number element 377
- select SQL statements executed monitor element 367
- self-describing data stream
  - database system monitor 7
  - event monitors 71
  - snapshot monitor 41
  - system monitor switches 19
- sequences
  - monitor elements
    - progress\_seq\_num element 415
    - sequence\_no 187
    - sequence\_no\_holding\_lk 317
- servers
  - monitor elements
    - product\_name 171
    - server\_instance\_name 168
    - server\_platform 171
    - server\_prdid 169
    - server\_version 170
- service-level information
  - monitor elements
    - service\_level 170
- session authorization ID 189
- shared workspace
  - health indicators
    - db.shrworkspace\_hitratio 569
  - monitor elements
    - shr\_workspace\_num\_overflows 280
    - shr\_workspace\_section\_inserts 281
    - shr\_workspace\_section\_lookups 280
    - shr\_workspace\_size\_top 279
- snapshot monitoring
  - administrative views 80
  - API request types 35
  - capturing
    - using SQL with file access 26
  - capturing to file 24
  - CLP commands 31
  - description 21
  - interpreting output for data partitions 84
  - making snapshot data available for all users 24
  - on data partitions 84

- snapshot monitoring (*continued*)
  - on partitioned database systems 40
  - output
    - samples 37
    - self-describing data stream 41
  - overview 3
  - request types 31
  - SQL table functions 27
  - subsections
    - subsection snapshots 39
  - using a client application 34
  - using CLP 31
  - using SQL 30
  - using SQL with direct access 22
  - with SNAP\_WRITE\_FILE 24
- snapshot time monitor element 406
- snapshots
  - capturing
    - using SQL with file access 26
  - capturing to file 24
  - capturing with SNAP\_WRITE\_FILE 24
  - making snapshot data available for all users 24
  - monitor elements
    - time\_stamp element 406
  - SQL table functions 27
  - using SQL with direct access 22
- sort overflows monitor element 224
- Sort Private Heap High Watermark monitor element 225
- sort share heap currently allocated monitor element 226
- Sort Share Heap High Watermark monitor element 226
- sorting
  - health indicators
    - db2.sort\_privmem\_util 558
  - monitor elements 219
    - active\_sorts 225
    - db.spilled\_sorts 560
    - pipeds\_sorts\_accepted element 222
    - pipeds\_sorts\_requested element 221
    - post\_shrthresold\_sorts monitor element 221
    - post\_threshold\_sorts element 220
    - sort\_heap\_allocated element 220
    - sort\_heap\_top monitor element 225
    - sort\_overflows element 224
    - sort\_shrheap\_allocated monitor element 226
    - sort\_shrheap\_top monitor element 226
    - total\_sorts element 222
- SQL communications area (SQLCA)
  - monitor elements
    - sqlca 383
- SQL dynamic statement text monitor element 381
- SQL operations
  - monitor elements
    - elapsed\_exec\_time element 454
- SQL requests since last commit monitor element 372
- SQL statements
  - displaying help 582
  - monitor elements
    - ddl\_sql\_stmts element 368
    - dynamic\_sql\_stmts element 364
    - failed\_sql\_stmts element 364
    - insert\_sql\_stmts element 461
    - num\_compilation element 397
    - num\_executions element 397
    - select\_sql\_stmts element 367
    - sql\_chains 434
    - sql\_reqs\_since\_commit element 372
    - sql\_stmts 433
- SQL statements (*continued*)
  - monitor elements (*continued*)
    - static\_sql\_stmts element 363
    - stmt\_pkgcache\_id element 389
    - stmt\_query\_id element 388
    - stmt\_sorts element 381
    - stmt\_source\_id element 388
    - stmt\_text element 381
    - stmt\_value\_data element 390
    - stmt\_value\_index element 391
    - stmt\_value\_isnull element 390
    - stmt\_value\_type element 390
    - total\_exec\_time element 398
    - uid\_sql\_stmts element 367
  - SQL table functions
    - capturing health snapshots 522
    - health monitor 574
  - SQL workspaces
    - monitor elements 279
  - sql\_chains element 434
  - sql\_stmts element 433
  - start stripe monitor element 341
  - statement best preparation time monitor element 398
  - statement first use time monitor element 385
  - statement history identifier monitor element 385
  - statement history list size monitor element 391
  - statement invocation identifier monitor element 387
  - statement isolation monitor element 386
  - statement last use time monitor element 386
  - statement nesting level monitor element 387
  - statement node monitor element 372
  - statement operation monitor element 374
  - statement package cache identifier monitor element 389
  - statement query identifier monitor element 388
  - statement sorts monitor element 381
  - statement source identifier monitor element 388
  - statement type monitor element 374
  - statement worst preparation time monitor element 398
  - statements
    - monitor elements
      - prep\_time\_best element 398
      - prep\_time\_worst element 398
      - stmt\_first\_use\_time element 385
      - stmt\_history\_id element 385
      - stmt\_history\_list\_size element 391
      - stmt\_invocation\_id element 387
      - stmt\_isolation element 386
      - stmt\_last\_use\_time 386
      - stmt\_nest\_level element 387
      - stmt\_node\_number element 372
      - stmt\_type element 374
- STATEMENTS event type
  - overview 45
- states
  - health indicators
    - db.alert\_state 562
    - db.db\_op\_status 562
    - db2.db2\_op\_status 561
    - ts.ts\_op\_status 558
- static SQL statements attempted monitor element 363
- statistics collection
  - health indicators
    - db.tb\_runstats\_req 563
- status
  - monitor elements
    - appl\_status 181
    - db\_status element 175

- status (*continued*)
  - monitor elements (*continued*)
    - db2\_status element 171
    - dcs\_appl\_status element 435
    - ss\_status element 393
- stmt\_operation element 374
- storage paths
  - monitor elements
    - num\_db\_storage\_paths 177
- stored procedure time monitor element 469
- stored procedures
  - monitor elements
    - stored\_proc\_time element 469
    - stored\_procs element 464
- stored procedures monitor element 464
- stripe set number monitor element 339
- stripe sets
  - monitor elements
    - container\_stripe\_set monitor element 338
- subsection execution elapsed time monitor element 393
- subsection node number monitor element 392
- subsection number monitor element 392
- subsection snapshots 39
- subsection status monitor element 393
- subsections
  - snapshot monitoring
    - subsection snapshots 39
- summary
  - health indicators 551
- SYSMON authority 21
- system monitor 3
- system monitor switches
  - description 15
  - self-describing data stream 19
  - setting from a client application 18
  - setting from the CLP 17
  - types 15
- System monitoring guide and reference
  - overview xi

## T

- table event monitors
  - creating 49
  - table management 52
- table file ID monitor element 351
- table name monitor element 343
- table queues
  - monitor elements
    - tq\_cur\_send\_spills 395
    - tq\_id\_waiting\_on 397
    - tq\_max\_send\_spills 396
    - tq\_node\_waited\_for 394
    - tq\_rows\_read 395
    - tq\_rows\_written 396
    - tq\_tot\_send\_spills 394
    - tq\_wait\_for\_any 394
- table reorganization
  - monitor elements 355
  - reorg\_end element 358
- table reorganize attribute flag monitor element 355
- table reorganize completion flag monitor element 358
- table reorganize end time monitor element 358
- table reorganize phase start time monitor element 356
- table reorganize start time monitor element 358
- table reorganize status monitor element 355
- table schema name monitor element 344

- table spaces
  - health indicators
    - ts.ts\_auto\_resize\_status 555
    - ts.ts\_op\_status 558
    - ts.ts\_util 556
    - ts.ts\_util\_auto\_resize 556
    - tsc.tscont\_op\_status 558
    - tsc.utilization 557
  - monitor elements
    - activity 320
    - bp\_tbsp\_use\_count 268
    - quiescer activity 335
    - quiescer\_ts\_id 336
    - range status 339
    - reorg\_long\_tbsp\_id 359
    - reorg\_tbsp\_id 359
    - tablespace\_auto\_resize\_enabled 332
    - tablespace\_content\_type 322
    - tablespace\_cur\_pool\_id 324
    - tablespace\_current\_size 332
    - tablespace\_extent\_size 323
    - tablespace\_free\_pages 326
    - tablespace\_id 320
    - tablespace\_increase\_size 333
    - tablespace\_increase\_size\_percent 334
    - tablespace\_initial\_size 332
    - tablespace\_last\_resize\_failed 335
    - tablespace\_last\_resize\_time 334
    - tablespace\_max\_size 333
    - tablespace\_min\_recovery\_time 330
    - tablespace\_name 320
    - tablespace\_next\_pool\_id 324
    - tablespace\_num\_containers 331
    - tablespace\_num\_quiescers 329
    - tablespace\_num\_ranges 331
    - tablespace\_page\_size 323
    - tablespace\_page\_top 326
    - tablespace\_pending\_free\_pages 326
    - tablespace\_prefetch\_size 323
    - tablespace\_rebalancer\_extents\_processed 328
    - tablespace\_rebalancer\_extents\_remaining 328
    - tablespace\_rebalancer\_last\_extent\_moved 328
    - tablespace\_rebalancer\_mode 327
    - tablespace\_rebalancer\_priority 329
    - tablespace\_rebalancer\_restart\_time 327
    - tablespace\_rebalancer\_start\_time 327
    - tablespace\_state 322
    - tablespace\_state\_change\_object\_id 330
    - tablespace\_state\_change\_ts\_id 330
    - tablespace\_total\_pages 325
    - tablespace\_type 321
    - tablespace\_usable\_pages 325
    - tablespace\_used\_pages 325
    - tablespace\_using\_auto\_storage 331
    - ts\_name 319
- table type monitor element 342
- tables
  - monitor elements
    - table\_file\_id element 351
    - table\_name element 343
    - table\_schema element 344
    - table\_type element 342
  - schemas
    - table\_schema element 344
- TABLES event type
  - overview 45

- TABLESPACES event type
  - overview 45
- terms and conditions
  - use of publications 586
- territory codes
  - monitor elements
    - territory\_code 195
- threads
  - monitor elements
    - agent\_pid 204
- thresholds
  - monitor elements
    - num\_threshold\_violations 484
    - threshold\_action 493
    - threshold\_domain 493
    - threshold\_maxvalue 494
    - threshold\_name 494
    - threshold\_predicate 494
    - threshold\_queuesize 495
    - thresholdid 495
  - threshold-based health indicators 511, 547
- time
  - monitor elements
    - prefetch\_wait\_time element 264
    - prep\_time 486
    - progress\_start\_time element 416
    - ss\_exec\_time element 393
    - stmt\_elapsed\_time element 380
    - time\_completed 496
    - time\_created 496
    - time\_of\_violation 496
    - time\_started 497
    - total\_sort\_time element 223
  - time waited for prefetch monitor element 264
  - time zone displacement monitor element 172
- time zones
  - monitor elements
    - time\_zone\_disp element 172
- timestamps
  - monitor elements
    - activate\_timestamp 470
    - db\_conn\_time 174
    - db2start\_time 168
    - last\_backup 177
    - last\_reset 405
    - lock\_wait\_start\_time 315
    - message\_time 412
    - prev\_uow\_stop\_time 201
    - statistics\_timestamp 492
    - status\_change\_time 183
    - stmt\_start 378
    - stmt\_stop 379
    - uow\_start\_time 201
    - uow\_stop\_time 202
- tokens
  - monitor elements
    - consistency\_token monitor element 376
    - corr\_token monitor element 193
- total fcm buffers received monitor element 234
- total fcm buffers sent monitor element 234
- total hash loops monitor element 229
- total log available monitor element 290
- total log space used monitor element 289
- total number of pages in object monitor element 357
- total progress work units monitor element 416
- total sort heap allocated monitor element 220
- total sort time monitor element 223

- total sorts monitor element 222
- transaction ID monitor element 454
- transaction processing monitors
  - monitor elements
    - tpmon\_acc\_str 460
    - tpmon\_client\_app 459
    - tpmon\_client\_userid 459
    - tpmon\_client\_wkstn 459
- transactions
  - monitor elements
    - num\_indoubt\_trans 312
    - xid monitor element 454
- TRANSACTIONS event type
  - overview 45
- troubleshooting
  - online information 586
  - tutorials 586
- tutorials
  - problem determination 586
  - troubleshooting 586
  - Visual Explain 585
- type at monitored (server) node monitor element 169

## U

- units of work (UOW)
  - monitor elements
    - prev\_uow\_stop\_time 201
    - progress\_total\_units element 416
    - uow\_comp\_status 203
    - uow\_elapsed\_time 202
    - uow\_id 497
    - uow\_lock\_wait\_time 315
    - uow\_log\_space\_used 289
    - uow\_start\_time 201
    - uow\_status 203
    - uow\_stop\_time 202
  - unread prefetch pages monitor element 264
  - update response time monitor element 467
  - update\_time element 467
  - update/insert/delete SQL statements executed monitor element 367
- updates
  - DB2 Information Center 584
  - monitor elements
    - update\_sql\_stmts element 462
- updates monitor element 462
- user authorization level monitor element
  - authorizations
    - authority\_lvl element 196
- utilities
  - monitor elements
    - utility\_description element 414
    - utility\_id element 412
    - utility\_invoker\_type element 414
    - utility\_priority element 413
    - utility\_type 413
  - utility description monitor element 414
  - utility id monitor element 412
  - utility invoker type monitor element 414
  - utility priority monitor element 413
  - utility start time monitor element 413
  - utility state monitor element 414
  - utility\_dbname element 412
  - utility\_start\_time element 413
  - utility\_state element 414

## V

- value data monitor element 390
- value has null value monitor element 390
- value index monitor element 391
- value type monitor element 390
- version monitor element 408
- version of monitor data monitor element 408
- Visual Explain
  - tutorial 585

## W

- watermark monitor elements
  - concurrent\_act\_top 474
  - concurrent\_connection\_top 475
  - concurrent\_wlo\_act\_top 475
  - concurrent\_wlo\_top 475
  - coord\_act\_lifetime\_top 477
  - cost\_estimate\_top 478
  - rows\_returned\_top 489
- watermarks
  - monitor elements
    - temp\_tablespace\_top 492
- Windows Management Instrumentation (WMI)
  - DB2 database system integration 99
  - description 99
- Windows operating systems
  - Performance Monitor
    - description 101
    - registering DB2 101
- WMI (Windows Management Instrumentation)
  - see Windows Management Instrumentation (WMI) 99
- workload management
  - monitor elements 470
- workloads
  - monitor elements
    - wlo\_completed\_total 498
    - workload\_id 500
    - workload\_name 500
    - workload\_occurrence\_id 501
- write-to-table event monitors
  - buffering 59

## X

- XDA Object Pages monitor element 354
- xda\_object\_pages monitor element 354
- xquery\_stmts monitor element 373





Printed in USA

SC23-5865-01





Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

**System Monitor Guide and Reference**

